Escola d'Enginyeria de Telecomunicació i Aeroespacial de Castelldefels
UNIVERSITAT POLITÈCNICA DE CATALUNYA

Trabajo fin de carrera

TITULO DEL TFC: Gas Turbine Design & Analysis Tool: Turbomachinery Components

TITULACIÓN: Ingeniería Técnica Aeronáutica, especialidad Aeronavegación

AUTOR: David Garcia Soto

DIRECTOR: Fernando Pablo Mellibovsky Elstein

FECHA: 9 de julio de 2012

Título: Gas Turbine Design & Analysis Tool: Turbomachinery Components

Autor: David Garcia Soto

Director: Fernando Pablo Mellibovsky Elstein

Fecha: 9 de julio de 2012

Resumen

Este proyecto se centra en la creación de un programa el cual ha de ser una herramienta de ayuda en el diseño y simulación de aplicaciones de motores de turbina de gas, permitiendo al usuario realizar los cálculos necesarios para poder implementar posteriormente los resultados en sus diseños. Por ello, dicha herramienta debe ser versátil y permitir trabajar tanto con un solo módulo como con diversos acoplados según las necesidades del usuario. Además el hecho de modularizar permite acotar y realizar de manera más efectiva tanto la modelización como la programación de los componentes. En este proyecto dichos módulos estudiados serán aquellos relacionados con la turbomaquinaria contenida en un motor de turbina de gas.

Para modelizar con éxito dichos componentes se harán necesarias ciertas bases teóricas que nos irán introduciendo a la materia a simular, tanto bases termodinámicas, mecánicas, numéricas como de programación. De este modo, se pretende no solo centrarse en aquellos aspectos únicamente relevantes para la simulación realizada, sino dar una visión más amplia y dejar, de modo teórico, maneras de implementar futuros módulos y ampliaciones más complejas. Por tanto, se podrá empezar en la versión más sencilla con un proceso termodinámico simple y llegar hasta versiones basadas en la simulación numérica directa (DNS), pasando por la introducción de eficiencias debido a pérdidas o la implementación de datos empíricos.

En el proyecto también se hará un estudio de los resultados obtenidos en el programa, comparándolos con otras fuentes que contengan dichos resultados. De este modo se permitirá valorar los progresos y la eficiencia del programa. Por otra parte, este tipo de comparaciones permitirá detectar errores que puedan surgir, bien sean debidos a la programación o conceptuales, permitiendo corregirlos o justificarlos si fuera necesario.

Por ultimo, es necesario decir, la verdadera magnitud del proyecto faltaría complementarlo con los módulos de aportación y substracción de calor y los de conducción, difusión y expansión de gases, realizados de manera simultánea en los proyectos *Gas Turbine Design & Analysis Tool: Ducting Components* y *Gas Turbine Design & Analysis Tool: Heat Transfer Components*.

Title: Gas Turbine Design & Analysis Tool: Turbomachinery Components

Author: David Garcia Soto

Director: Fernando Pablo Mellibovsky Elstein

Date: July, 9th 2012

Overview

This project focuses in the creation of a program which will be a tool for helping in the design and simulation for gas turbine applications, allowing to the user to perform later the results to his designs. That's why this tool must be versatile and allows working with a single module or a group of them in function of the user necessities. Also the fact of modularize allows to delimitate and perform effectively as well the modeling and the programing of the components. In this project this studied modules will be related with turbomachinery inside a gas turbine.

For modeling successfully these components it will be necessary some theoretical bases for introduce us to the subject to simulate like thermodynamics, mechanics, numerical or programming bases. In this way, the desire is to not only focus on these more relevant aspects of the simulation being carried out, but give a broader view and leave, in a theoretical way, future possibilities to implement more complex modules and extensions. So, will be started whit a simple thermodynamic process in the first version and arrive to direct numerical simulations (DNS) in latest versions, incorporating efficiencies caused because of the losses or implementation of empiric data.

In this project will be done a study of the obtaining results in the program, and then those results will be compared with other references that contain similar final values. So, this will allow us to know the progress and efficiency of program. Also, this kind of comparisons will be a way to detect possible errors that can appear. These errors could be in the program or can be theoretical, any way, we must to correct or justify them if it's needed.

Finally, it must be said, that this final year project is only a small part of a larger project that aims to fully simulate the behavior of a gas turbine engine in the most realistic way possible. So, in order to see the full scale of the project, it remains to be completed with heat transfer and subtracting modules and conduction, diffusion and expansion of the gases modules. These modules will be done in simultaneously in the projects named *Gas Turbine Design & Analysis Tool: Ducting Components* and *Gas Turbine Design & Analysis Tool: Heat Transfer Components*.

Dedico esté trabajo a Fernando por su interes y ayuda para que éste trabajo pudiera ser posible. También a mis compañeros Ferran y Jóse por su aportación al trabajo con sus módulos. Finalmente, agradecer a la família y amigos por el soporte y los animos que me han dado.

INDICE

INTRODUCCIÓN

En este trabajo se pretende emular en diferentes grados el comportamiento de los elementos mecánicos móviles que contendría un motor de turbina de gas. Estas partes se conocen como turbomaquinaria, cuya definición más genérica engloba todas aquellas máquinas que transfieren energía entre un rotor y un fluido.

Con el propósito de simular estos componentes utilizaremos varios desarrollos matemáticos, principios termodinámicos y aerodinámicos los cuales aplicaremos a la programación en C++. Dicha programación será progresiva y por tanto se distinguirán entre diversos grados de modelización, desde el más sencillo hasta el más complejo, de este modo se irán asentando mejor los conceptos y analizando todas las posibilidades, de este modo el resultado final será completo y robusto.

Se hace necesario remarcar que en este trabajo se usaran en todo momento unidades definidas por el SI (Sistema Internacional), basado en los metros, los segundos y los quilogramos. En el caso que no se aplicara este sistema se especificará de manera adecuada.

El trabajo se distribuirá de manera de que resulte progresivo en el desarrollo de los conceptos y se vaya siguiendo la teoría a medida que se va avanzando en los grados de modelización. Por este motivo, se iniciara el trabajo con un capítulo con conceptos previos en el cual se definirá el concepto de turbomaquinaria y se tratarán aquellos temas que vayan a ser usados en los diversos grados de modelización.

Los siguientes capítulos intentaran seguir una pauta común, en la cual primeramente se explicaran los conceptos teóricos nuevos para el módulo que corresponda. Seguidamente un apartado de programación, donde se expliquen aquellos conceptos necesarios para llevarla acabo, así como que información o detalles se tienen que tener en cuenta. Finalmente, se trataran los resultados, explicando como obtenerlos y si se corresponden con lo deseado.

Con el propósito de testear los módulos se recurrirá al *Gas Turb 11* que es un programa referencia para el nuestro, pues se trata también de una herramienta para la ayuda en el diseño de motores de turbina. En este se pueden encontrar varios modelos de motor ya creados, con la opción de poder cambiarles determinados valores que podrían interesar en el diseño del usuario. En este trabajo lo podemos usar de guía ya que buscamos algunos parámetros comunes.

Conociendo de la existencia de dicho programa, se debe indicar que diferenciación ofrece el nuestro con respecto a éste u otros existentes. Pues bien, la principal diferencia se atribuye a la modularización de los componentes, es decir, nuestro programa ofrecerá más posibilidades aparte de algunos ejemplos de motores creados, como la posibilidad de testear los módulos por

separado o bien, crear con los módulos deseados y en el orden que se quiera todo tipo de motores.

En los últimos capítulos se explicará como interactúan con otros módulos pertenecientes a los motores de turbina de gas. Para ello se vera un ejemplo de turbo reactor y se analizaran brevemente sus resultados.

Para terminar se dejará constancia de cuales deberían ser los pasos a seguir para futuros desarrollos de dicha herramienta de simulación, permitiendo grados de realismo superior.

CAPÍTULO 1. CONCEPTOS PREVIOS

Para poder comprender mejor el contenido de este trabajo es necesario un capítulo previo para situarnos en el tema. En este capítulo encontraremos definiciones y explicaciones teóricas de algunos conceptos relevantes, conectados sencillamente con el funcionamiento genérico de un motor de turbina o bien más de base termodinámica.

1.1. Motor de turbina de gas

Las turbinas son máquinas motrices de flujo continuo que producen trabajo mecánico mediante un sistema de alabes de formas diversas empleando la energía cinética, térmica o de presión de un fluido. Por tanto, las turbinas de gas emplean para su funcionamiento elementos gaseosos, mayoritariamente el aire.

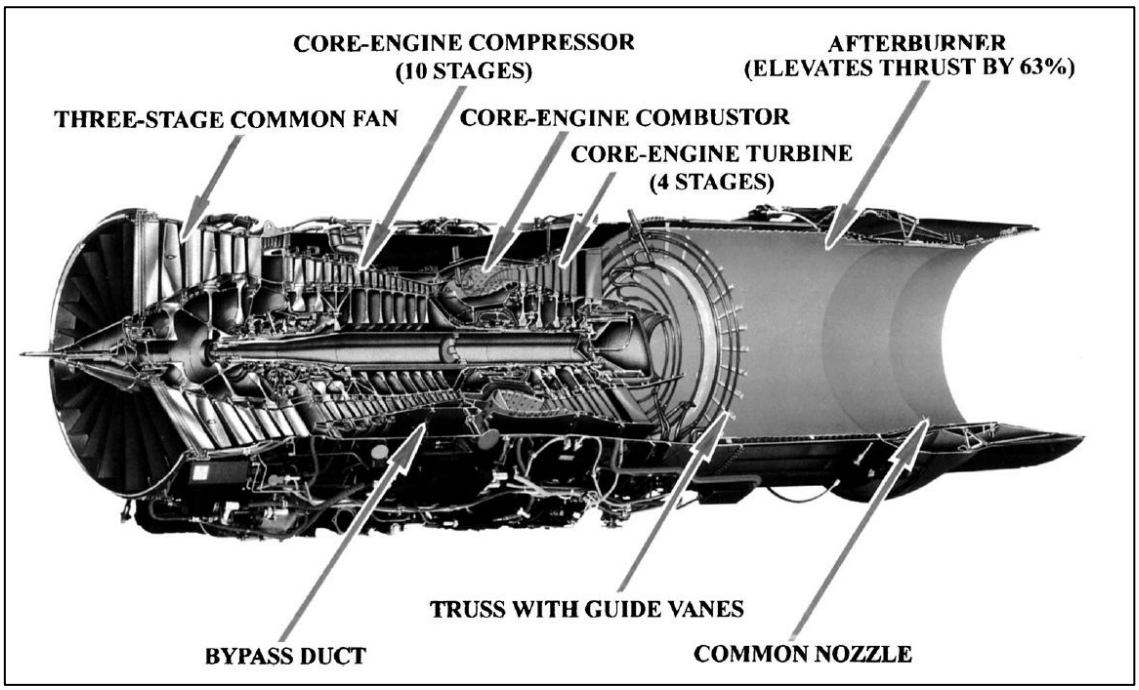


Fig. 1.1 Sección de un turbo reactor como ejemplo de turbina de gas y los diferentes componentes. Imagen extraída de la referencia [3]

El funcionamiento de los motores de turbina de gas se basa en el ciclo termodinámico de Joule. El motor aspira aire de la atmósfera y un compresor lo comprime y lo envía a una cámara donde se inyecta combustible, que arde de forma continua y eleva la temperatura del fluido. Detrás del generador de gas se encuentra la turbina propiamente dicha, unida directamente al compresor mediante un eje.

Si la propulsión es por reacción, la turbina tiene por única misión de arrastrar el compresor. Los gases de escape se aceleran en una tobera y son expulsados a gran velocidad; la variación de la cantidad de movimiento del fluido entre la entrada del motor y la salida de la tobera produce un empuje hacia delante que se emplea para la propulsión.

Si la propulsión es por acción, la turbina produce la expansión casi hasta la presión atmosférica, recibiendo más trabajo del que es necesario para accionar el compresor. El trabajo sobrante se transmite a través de un reductor.

El primer tipo de motor es usado mayoritariamente para la propulsión de aeronaves a altas velocidades en el campo aeronáutico. Mientras que el segundo es más usado en la industria y en la propulsión de vehículos de propulsión mecánica, como coches, barcos o aviones de turbohélice.

1.2. Turbomaquinaria

La turbomaquinária se puede definir como todo aquel conjunto de máquinas que transfieren, mediante un trabajo mecánico, energía entre un rotor y un fluido. Remarcar, que como consecuencia de este trabajo son inherentes a estas máquinas unas perdidas mecánicas.

Para los motores de turbina de gas estos elementos serán fundamentalmente las turbinas y los compresores, pudiendo añadir varias etapas y un compresor conocido como fan. Mientras que una turbina transfiere energía de un fluido a un rotor, un compresor transfiere energía desde un rotor para un fluido. Los dos tipos de máquinas se rigen por las mismas relaciones básicas como segunda ley de Newton del movimiento y la ecuación energía de Euler para fluidos compresibles.

Otro tipo de turbomáquinas, por ejemplo, son las bombas centrífugas en las que la transferencia de energía se produce desde el rotor hasta el fluido, normalmente un líquido; mientras que las turbinas y compresores generalmente trabajan con un gas.

1.3. Componentes

En este apartado, se describirán con más detalle aquellos componentes más característicos de la turbomaquinaria de un motor de turbina de gas.

1.3.1. Compresor

Son aquellas máquinas encargadas de comprimir el gas de forma continua, aumentando la energía del fluido debido a la presión, mediante un trabajo

mecánico. Uno de los principales parámetros que lo definen es la relación de compresión que nos permite saber su capacidad de comprimir.

Según la manera en que circula el flujo másico de aire a través de dicho componente se hará la distinción entre los dos tipos de compresores existentes: los compresores centrífugos y los axiales.

2.1.1.1. Compresor centrifugo

El compresor centrífugo consiste fundamentalmente en una carcasa inmóvil que contiene en su interior un rodete que al girar le da al aire un incremento de velocidad considerable. Aparte hay una serie de conductos divergentes fijos en los cuales el aire se frena lo que consecuentemente implica un aumento de la presión estática, por este motivo es importante definir bien los parámetros estáticos que se verán en el capitulo 3. Estos conductos implican un proceso de difusión por ello se les denominara difusor, pero en ningún caso deben confundirse con el difusor genérico a la entrada de ciertos motores. En la figura 1.2 se puede ver dichas partes esquematizadas. La imagen 'a' muestra una vista desde delante/ alzado del difusor, mientras que 'b' y 'c' son el perfil para el caso en el que sea de una cara o de dos caras respectivamente. Aunque actualmente el más extendido es el de una sola cara.

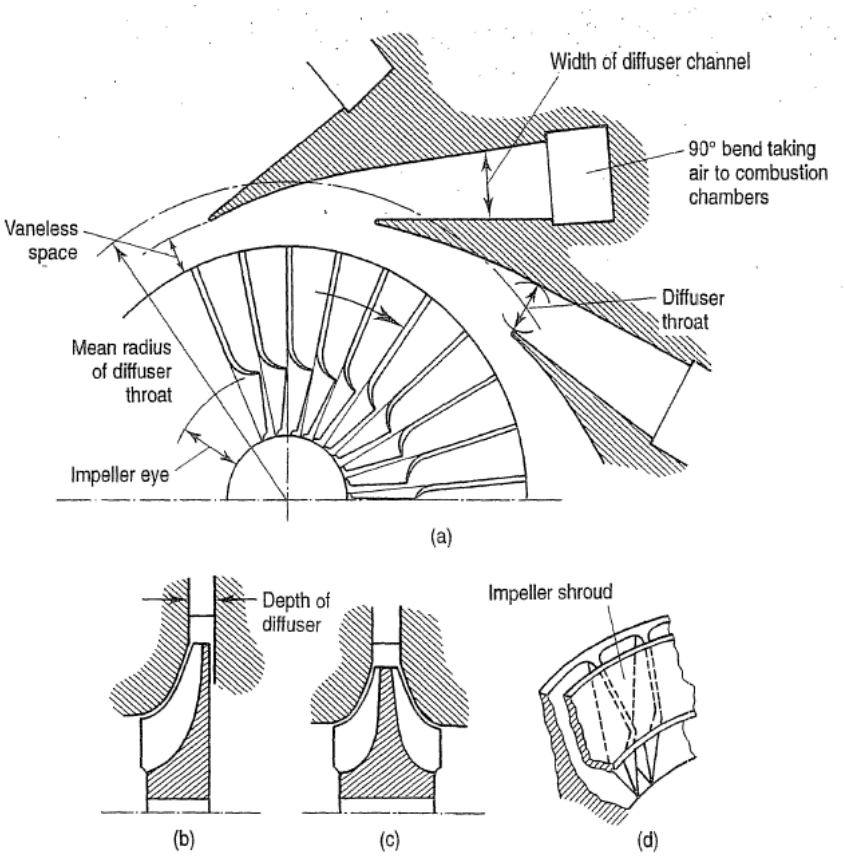


Fig. 1.2 Partes esquematizadas de un compresor centrífugo. Imágen extraída de la referencia [1]

Si bien es cierto que este tipo de compresor no es usado actualmente tan a menudo como antes en turbinas de gas, sigue siendo una opción a considerar si se requiere ocupar una longitud pequeña, o bien por que se desea trabajar en una atmosfera con cierto grado de impurezas que pudieran dañar fácilmente uno de tipo axial, pero sin duda un aspecto relevante es que puede trabajar con buen rendimiento para un amplio margen de flujos másicos con cualquier velocidad de giro.

2.1.1.2. Compresor axial

Este tipo de compresores esta compuesto por alabes distribuidos entre el rotor y el estator. Los alabes de rotor serán móviles mientras que los del estator permanecerán fijos, los cuales recuperaran parte de la energía cinética del aire en forma de un aumento de presión, aparte también se encargaran de dirigir al flujo con el ángulo adecuado par su entrada en la siguiente etapa de alabes móviles. Por tanto se puede deducir que el compresor axial esta formado por una serie de rotores y estatores intercalados entre si. A cada par de rotor y estator se los denominara escalonamiento.

A diferencia del caso anterior con los centrífugos, este tipo de compresor permitirá tener un reducido ancho, pero por el contrario esto representará una longitud mayor. Este reducido tamaño de ancho puede resultar sumamente beneficioso sobretodo para los motores destinados a la aviación, puesto que aerodinámicamente ablando supone una superficie menor de resistencia aerodinámica. Si se compara una sola etapa veremos que la relación de compresión es menor que en los centrífugos, pero como ventaja, si se unen diversas etapas, se supera la relación de los compresores centrífugos. Por ultimo, decir que son mas difíciles de fabricar y por tanto que su coste es mayor.

1.3.2. Turbina

Las turbinas son aquellos mecanismos giratorios que extraen energía de una corriente de gas, cambiando sus condiciones termodinámicas. El gas pierde energía potencial convirtiéndose en energía de rotación para el eje de dicha turbina, que normalmente servirá para auto-mantener la rotación del compresor. Por otra parte, los gases sobrantes de la expansión serán usados para la propulsión u otras aplicaciones.

2.1.1.3. Turbinas axiales y centrípetas o radiales

Las turbinas centrípetas se asemejan a los compresores centrífugos, las diferencias recaen en que las turbinas dirigen el flujo hacia dentro y los alabes son de tobera en vez de difusor. Hay que puntualizar que este tipo de componente no es frecuentemente usado en motores a reacción a menos que interese un motor corto y rígido. Esto es debido a que posee un rendimiento

muy inferior a la turbina axial y no es demasiado apta para las altas temperaturas de estos motores.

Para el caso de las turbinas axiales la diferencia principal con el compresor de este mismo tipo es el orden en las etapas. Puesto que se quiere unos resultados inversos al del compresor, el orden de los rotores y los estatores será inverso.

1.3.3. Fan

El fan es fundamentalmente un compresor, pero se hace una distinción del resto de etapas de compresor puesto que esta situado a la entrada del motor y la mayoría de las veces en él se produce una división de flujos. Por tanto, en grados de complejidad mayor se tendrá que diferenciar con el resto de compresores.

1.4. Programación orientada a objetos

Con el fin de mejorar y hacer más entendible el programa se hará necesario, como ya se verá, la programación orientada a objetos, es por ello que en este aparatado se dará una visón general de este concepto.

La programación orientada a objetos nos permitirá crear clases. Las clases, a su vez, permiten al programador modelar objetos que contienen miembros de datos y funciones de miembro, es decir, atributos y operaciones. Por tanto, en nuestro caso, nosotros definiremos una clase estableciendo como miembros de datos las posibles incógnitas y parámetros que definen cada componente de la turbomaquinária de la turbina de gas. Por otro lado, en las funciones de miembro encontraremos todas aquellas funciones necesarias para poder resolver un componente. Un ejemplo de esta definición de objeto la podemos ver en los anexos B que ira incluida en los ‘headers’ *(.h)* de los componentes. Esto significa que los atributos no serán directamente modificables por las operaciones.

Para inicializar los miembros de una clase correctamente debemos usar un constructor. Una función con el mismo nombre de la clase, pero precedido por un carácter tilde (~) llamará al destructor.

Por tanto, se puede ver que con ello ganamos claridad en el programa principal, así como facilidad para repetir funciones y crear y destruir componentes según sea requerido. Este tipo de programación incluida en esta etapa se mantendrá durante los demás grados de modelización intentando mejorarlo.

1.5. Métodos numéricos

Los métodos numéricos son técnicas de análisis mediante las cuales es posible formular problemas de tal manera que se puedan resolver utilizando operaciones aritméticas.

1.5.1. Método de Newton-Raphson

Para poder programar la turbomaquinária será necesario una manera de resolver las ecuaciones no lineales, por ello en este apartado se describirá el funcionamiento del método de Newton-Raphson de manera descriptiva para posteriormente explicar los algoritmos de programación y como se ha programado en los diferentes casos.

El método de Newton-Raphson se podría incluir en una subcategoría de métodos abiertos. Como un método abierto nos referimos a aquellos que requieren de uno o dos valores de la incógnita pero que no necesariamente este debe encerrar la raíz. La relativa cercanía del punto inicial a la raíz depende mucho de la naturaleza de la propia función, si ésta presenta múltiples puntos de inflexión o pendientes grandes en el entrono de la raíz, entonces las probabilidades de que el algoritmo diverja aumentan, lo cual exige seleccionar un valor supuesto cercano a la raíz.

$$x_{n+1} = x_n - \frac{fx_0}{\delta fx_0} \qquad \textbf{(1.1)}$$

Se ha de tener en cuenta que la formula (2.2) es solo aplicable en el caso de una sola variable. En la figura (2.3) se puede ver gráficamente un ejemplo de dicho método.

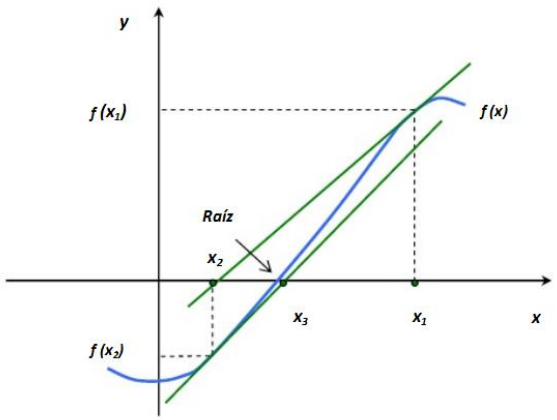


Fig. 1.2 Representación gráfica del método de Newton-Raphson. Gráfica extraída de la referencia [8].

Hay que tener en cuenta que el orden de convergencia de este método es, por lo menos, cuadrático. Sin embargo, si la raíz buscada es la multiplicidad mayor a uno el método de Newton perderá la convergencia cuadrática y pasará a ser lineal.

El motivo por el cual se ha escogido este método y no otros, como por ejemplo el método de la secante, es por su rapidez para converger en el resultado. Aunque hay la posibilidad de que no converja se seguirá optando por este método, debido a que para el grado de dificultad de las ecuaciones implementadas y definiendo bien los valores iniciales las posibilidades de que no converja son muy reducidas.

1.5.2. Explicación y programación del método de Newton-Raphson para funciones vectoriales de varias variables

Para este caso en el que tenemos varias funciones a resolver es imprescindible poder encontrar un sistema para programar el método. En este caso se creará un vector con las incógnitas, este vector tendrá la longitud del número de funciones a resolver. Una vez creado dicho vector se empezará a crear una matriz conocida como Jacobiano, la cual contiene las derivadas parciales de todas las funciones respecto de las distintas incógnitas, tal y como se muestra en la formula (1.4).

$$\begin{cases} f_1(x_1, \cdots, x_n) = f_1(\mathbf{x}) = 0 \\ \cdots\cdots\cdots \\ f_n(x_1, \cdots, x_n) = f_n(\mathbf{x}) = 0 \end{cases} \quad \textbf{(1.2)}$$

$$\mathbf{x} = [x_1, \cdots, x_n]^T \quad \textbf{(1.3)}$$

$$J_f(\mathbf{x}) = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \cdots & \frac{\partial f_1}{\partial x_n} \\ \cdots & \cdots & \cdots \\ \frac{\partial f_n}{\partial x_1} & \cdots & \frac{\partial f_n}{\partial x_n} \end{bmatrix} \quad \textbf{(1.4)}$$

Una vez completada dicha matriz se podrá implementar la fómula de Newton-Raphson para diversas variables mostrada a continuación.

$$\mathbf{x} \Leftarrow \mathbf{x} + \delta\mathbf{x} = \mathbf{x} - J_f^{-1}(\mathbf{x}) f(\mathbf{x}) \quad \textbf{(1.5)}$$

Para poder programar dicha fórmula harán falta unas librerías externas, puesto que para programar de manera eficiente y de manera rápida, su uso es la opción más adecuada. Concretamente las librerías LAPACK y BLAS son las usadas para este proyecto. La rutina que se utilizara será la dgsev, la cual se encargará de invertir la matriz Jacobiana y multiplicarla por la matriz de funciones. Una vez que las librerías externas han hecho dicho proceso solo se tendrán que restar dicho vector resultado de la operación al valor del vector de incógnitas de la iteración anterior.

Para acabar de dejar claro este concepto a continuación se presenta un diagrama de flujos para la programación con una sola incógnita. Este seriá extrapolable a varias incógnitas siempre y cuando se tenga en cuenta que el segundo paso se tendría que iterar tantas veces como número de incógnitas, que Δx es constante para dichas iteraciones, y que el tercer paso equivaldría a la creación de la matriz jacobiana. Los demás procedimientos se deben operar vectorialmente.

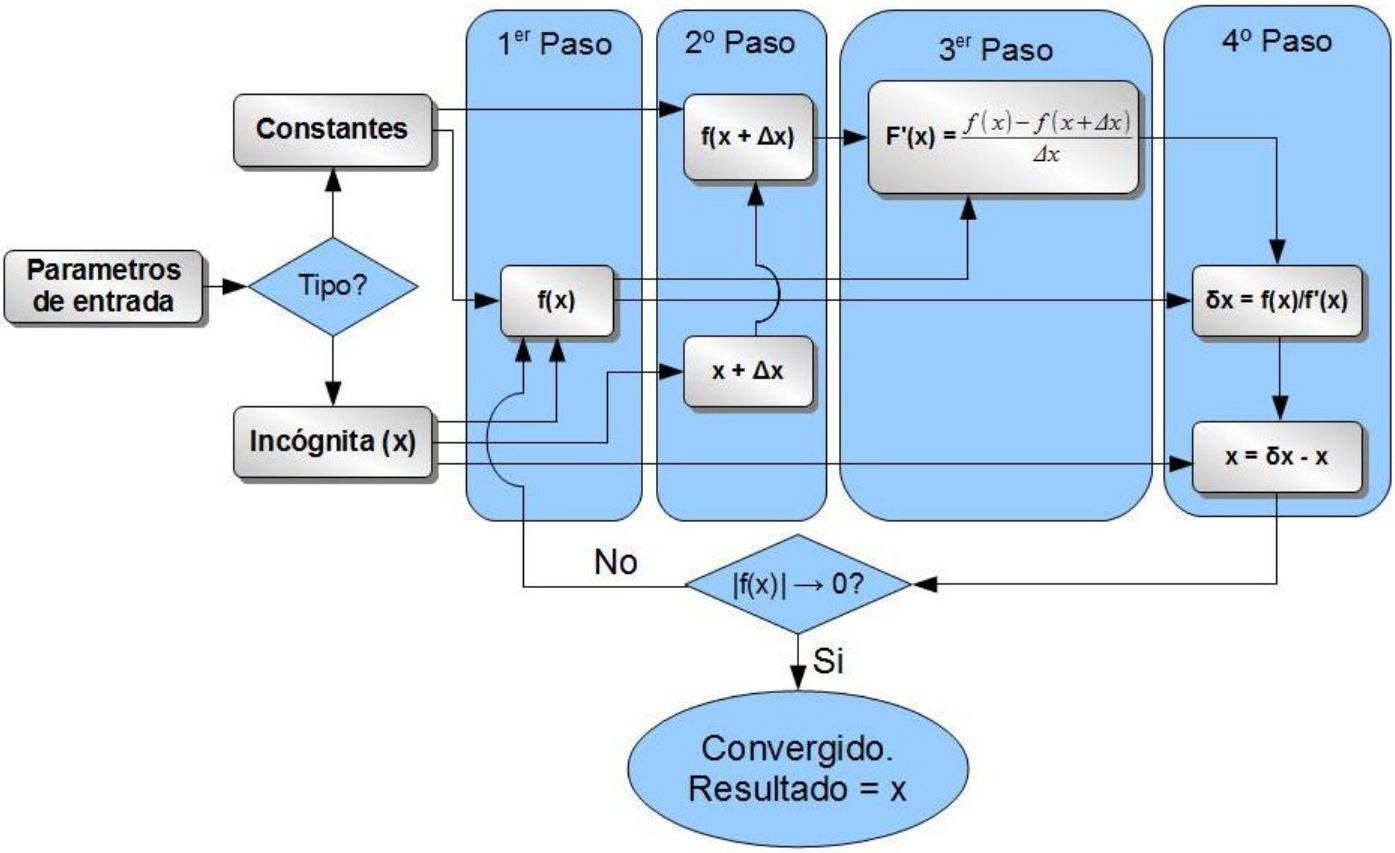


Fig. 1.3 Diagrama de flujos de la programación del Newton-Raphson para una sola incógnita.

CAPÍTULO 2. MODELIZACIÓN 0: TURBOMAQUINARIA IDEAL

En éste primer apartado se simulará la turomaquinária en función del ciclo simple de la turbina de gas, es decir, el ciclo de Brayton.

2.2. El ciclo de Brayton

Un proceso termodinámico es cualquier cambio de un estado de equilibrio a otro experimentado por un sistema, definido por un estado inicial y otro final, así como una trayectoria y las interacciones con los alrededores.

Si los estados a lo largo de la trayectoria de un proceso son de equilibrio y, por tanto, puede evolucionar espontáneamente en ambos sentidos, dicho proceso será reversible. Como consecuencia se podrá invertir sin dejar ningún rastro en los alrededores, es decir, tanto el sistema como los alrededores vuelven a sus estados iniciales una vez finalizado el proceso inverso. Esto es posible sólo si el intercambio de calor y trabajo netos entre el sistema y los alrededores es cero para el proceso combinado.

Los procesos adiabáticos son aquellos en los cuales no hay una transferencia de calor, bien sea porque el sistema está aislado como porque tanto el sistema como los alrededores estén a la misma temperatura y por tanto no sea necesaria dicha transferencia. Esté último caso solo sería válido si previamente se cumple la definición de proceso reversible. Otro punto de vista podría ser el hecho de que el ritmo de la transferencia de calor fuera negligible frente a los intercambios energéticos propios del proceso para altas velocidades.

Aquellos procesos en los cuales la entropía se mantiene constante serán procesos isentrópicos. Todos los procesos que sean adiabáticos y reversibles serán procesos isentrópicos. Muchos sistemas de ingeniería como bombas, turbinas, toberas y difusores se pueden modelizar siguiendo como patrón estos tipos de procesos.

Si el sistema regresa a su estado inicial al final del proceso, por tanto que sus características termodinámicas son idénticas tanto en el estado inicial como final, se estará definiendo un ciclo.

Por tanto según dichas definiciones, el ciclo de Brayton esta constituido por cuatro procesos reversibles:

1-2: Compresión isentrópica.
2-3: Adición de calor a presión constante.
3-4: Expansión isentrópica.
4-1: Rechazo de calor a presión constante.

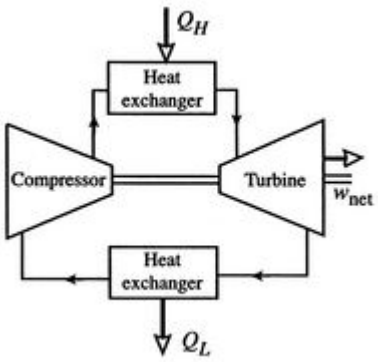


Fig. 2.1 Esquemático termodinámico del ciclo de Brayton

A pesar que en el esquemático anterior pueda parecer que el sistema es cerrado, realmente se trata de un sistema abierto, y por tanto se aplicaran las leyes de la termodinámica para sistemas abiertos.

Por otra parte, se analizaran los procesos suponiendo un flujo estacionario, dando lugar al diagrama de la figura (2.2). Los procesos de flujo estacionario son aquellos procesos durante los cuales un fluido fluye de forma estacionaria por un volumen de control. Es decir, las propiedades del fluido pueden cambiar de un punto a otro dentro del volumen de control, pero en algún punto fijo permanecen sin cambio durante todo el proceso.

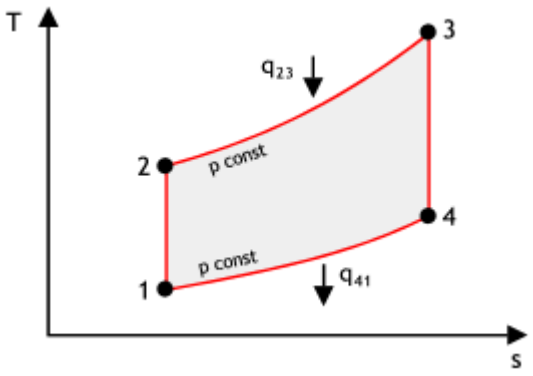


Fig. 2.2 Diagrama T-s del ciclo de Brayton

La ecuación de la energía para el flujo estacionario correspondiente a este ciclo será:

$$Q = (h_2 - h_1) + \frac{1}{2}(C_2^2 - C_1^2) + W \qquad \textbf{(2.1)}$$

Siendo Q el calor, W el trabajo específico, C1la velocidad inicial del fluido y C2 la velocidad final del fluido.

Si aplicamos la ecuación del ciclo de Brayton exclusivamente a la turbomaquinária, es decir, al proceso de compresión y de expansión del ciclo, se podrá obtener una primera modelización ideal y a la vez sencilla del compresor y la turbina. Para ello es necesario suponer que la variación de la energía cinética del gas entre la entrada y la salida de cada elemento es despreciable.

2.3. Programación de la turbomaquinária ideal

Para programar los módulos se empezará con uno de los elementos, el cual a estos niveles de modelización, será fácilmente extrapolable al resto de ellos. Empezaremos por hacerlo con una sola incógnita y posteriormente se complicará creando tantas variables como ecuaciones tenga el modulo.

2.3.1. Programación con una sola incógnita

El primer prototipo del programa se basa en el ciclo de Brayton. Para familiarizarse con la materia, el lenguaje de programación C++ y la manera de programar del *Microsoft Visual Studio 2010,* el primer prototipo de compresor se considera con una sola función de este componente. Definiendo como parámetros tres de las cuatro variables de la función, y la restante siendo la incógnita, se permite solucionar el problema de manera más comprensible y visible. De este modo, se tendrá un mejor dominio de todos los factores y permitirá detectar de manera más rápida posibles contratiempos que puedan surgir en la realización de posteriores grados de modelización más complejos.

La fórmula del compresor (2.2), escogida en este caso, relaciona la temperatura y presión de entrada con las de salida mediante un proceso adiabático e isentrópico.

$$P_i^{1-\gamma} T_i^{\gamma} = P_o^{1-\gamma} T_o^{\gamma} \qquad \textbf{(2.2)}$$

Un proceso politrópico es aquel en el que la relación $P(V)^n$ se mantiene contante, recordemos que PV=mRT, y por tanto si se sustituye y desarrolla un poco se obtiene $P^{1-n}\,T^n = constante/(m\,R)^n$ y puesto que m y R son constantes $P^{1-n}\,T^n$ seguirá siendo constante.

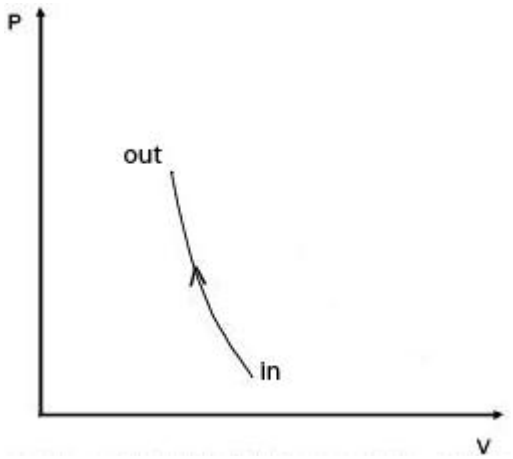


Fig. 2.4 Diagrama P-v del proceso politrópico en un compresor ideal.

Con el propósito de trabajar de una manera apropiada con la formula anterior (2.3) pasaremos a definirla como la función 1 del compresor (2.3), que deberá evaluar a cero cuando los valores de la temperatura y la presión en la entrada y la salida correspondan a una solución.

$$fx_1 = \left(P_o^{1-\gamma}\, T_o^{\gamma}\right) - \left(P_i^{1-\gamma}\, T_i^{\gamma}\right) \qquad \textbf{(2.3)}$$

Una vez definida la función se introducirán en ella los parámetros y la variable incógnita con un valor inicial (*incial guess*). Este valor inicial será una cantidad aproximada a una posible solución típica o realista. Por ejemplo, para la temperatura de entrada le asignaremos 288K correspondientes a la temperatura estándar a nivel del mar, asumiendo en este caso que el motor se encuentra en tierra.

Para poder aplicar el método de Newton se ha de evaluar la derivada de la función con respecto a la incógnita, para ello se usaran técnicas de diferencias finitas, explicado a continuación. Una vez se tiene un primer valor para la función se incrementará un delta de x, de entre 0.1 y 10^{-6}, al valor inicial y sustituyendo nuevamente fx_1 obtenemos fx_1' y entonces se podrá aplicar la fórmula (2.4) para obtener una aproximación a la primera derivada con respecto la incógnita deseada.

$$\delta fx_1 = \frac{fx_1' - fx_1}{\Delta x} \qquad \textbf{(2.4)}$$

A partir de este momento se podrá empezar a utilizar la ecuación propia del método, correspondiente a la fórmula (1.1) vista anteriormente, la cual describe de manera simple el método numérico de Newton-Raphson. Iterando dicha ecuación la función convergirá entregando una solución que satisfaga $|fx_1| \leq 0.00001$, o la resolución que se requiera.

Con tal de complementar este primer prototipo se pueden añadir las formulas (2.5) y (2.6). De este modo también se obtendrá una manera fiable de poder verificar en la siguiente modelización los valores del trabajo necesario para el funcionamiento del compresor y la relación de compresión.

$$r_c = \frac{P_o}{P_i} \quad \textbf{(2.5)}$$

$$W = (h_i - h_o) = C_p(T_o - T_i) \quad \textbf{(2.6)}$$

De manera análoga se programara la turbina puesto que en el ciclo de Brayton no se producen cambios en las formulas y conceptos que definen el componente. Por tanto nuevamente se aplicaran las fórmulas (2.3), (2.5) y (2.6) para programar la turbina.

A pesar de las similitudes, conviene remarcar que debido a su funcionamiento inverso, como se ve claramente en la figura 2.5, se obtendrán relaciones de compresión entre 0 y 1, así como conviene recordar que el trabajo subministrado por la turbina es contrario al del compresor, por tanto el signo que lo defina será inverso, en el caso de este trabajo negativo.

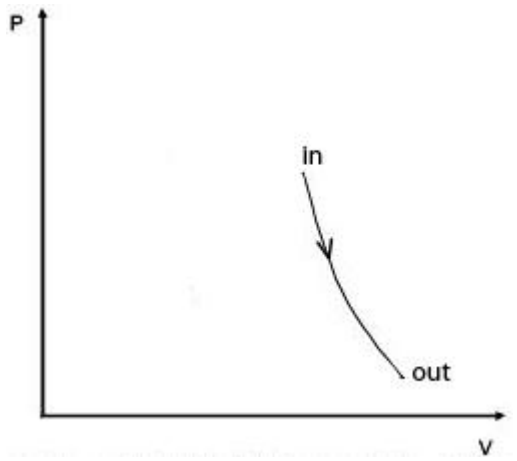


Fig. 2.5 Diagrama P-v del proceso politrópico en una turbina ideal.

2.3.1.1. Test de comprobación del Newton-Raphson con una sola incógnita

Como resultado inmediato de este primer prototipo se puede ver el código adjunto en los anexos B. También, se puede apreciar que parte de este código se encargará de mostrar en una interfaz de tipo *"Consola de Win 32"* que permitirá testear los componentes anteriores.

En este primer caso más sencillo, simplemente comprobaremos que el programa responda conforme lo esperado, así como asegurarse que calcula de forma correcta las fórmulas (2.5) y (2.6).

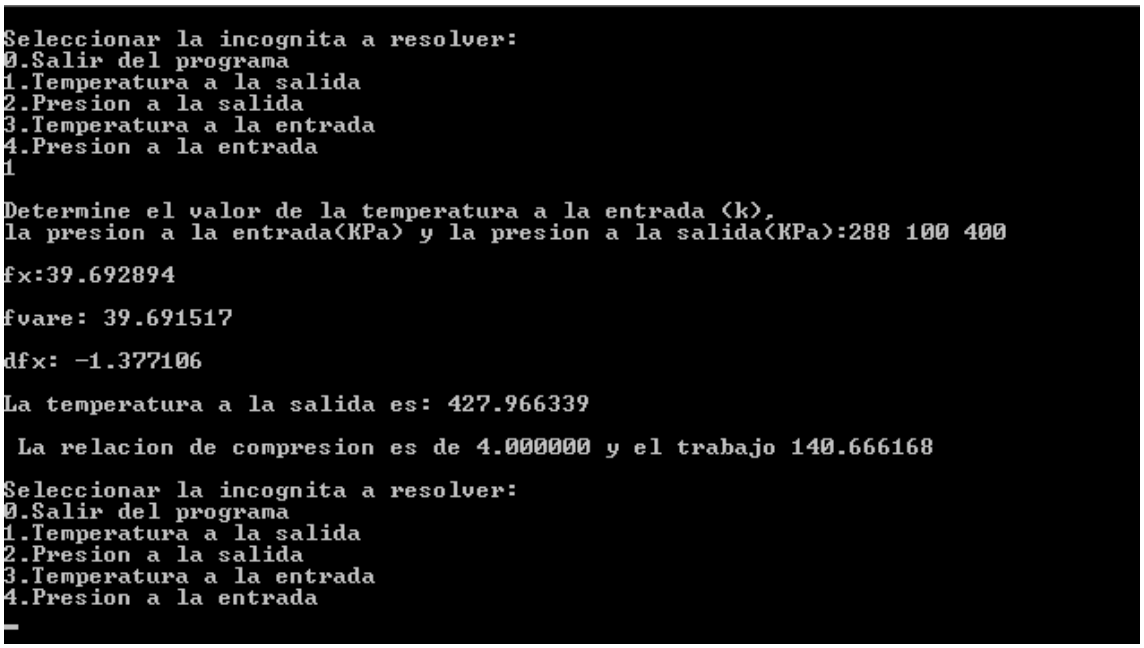


Fig. 2.6 Ejemplo de ejecución del primer prototipo

Por ejemplo, como se muestra en la figúra 2.6 la interfaz proporciona un menú para ejecutar uno de los casos, del 1 al 4, o bien salir, en el cao 0. Las opciones de la 1 a la 4 permitirán escoger la incógnita deseada, para el ejemplo: la temperatura de salida. Una vez definida, se pedirán el resto de parámetros y una vez introducidos se resolverá el compresor. Finalmente, el programa devuelve el valor de la temperatura convergido, junto la relación de compresión, el trabajo y, nuevamente, el menú.

Si ahora quisiéramos testear que los datos del programa son correctos simplemente se tendría que aplicar las formulas del apartado anterior, teniendo en cuenta un cp=1.005 y ɣ=1.4 y veremos como corresponden los valores. En este caso las unidades no seguirán el SI y el trabajo aparecerán en kJ, la presión en kPa y la temperatura en K. Aclarar que fvare corresponde al valor de fx'(x+Δx) y dfx corresponde a la derivada parcial con respecto de la incógnita.

2.3.2. Programación multi-variable

En este siguiente nivel de programación se pasará a complicar el desarrollo del código usando programación orientada a objetos, pero con ello se ganará claridad y eficiencia en el código. Además, al usar varias funciones, y por tanto, varias incógnitas será indispensable buscar una manera efectiva de aplicar el

método numérico de Newton-Raphson, dicha manera ya se ha explicado en la introducción.

Por tanto las funciones empleadas en el código para éste modulo, aparte de la primera ya definida en la fórmula (2.3), serán las siguientes:

$$fx_2 = P_{in} \cdot r_c - P_{out} \quad \textbf{(2.7)}$$

$$fx_3 = W - Cp * (T_{out} - T_{in}) \quad \textbf{(2.8)}$$

Al usar la programación orientada a objetos, se podrá definir el objeto compresor 0 siguiendo el esquema siguiente:

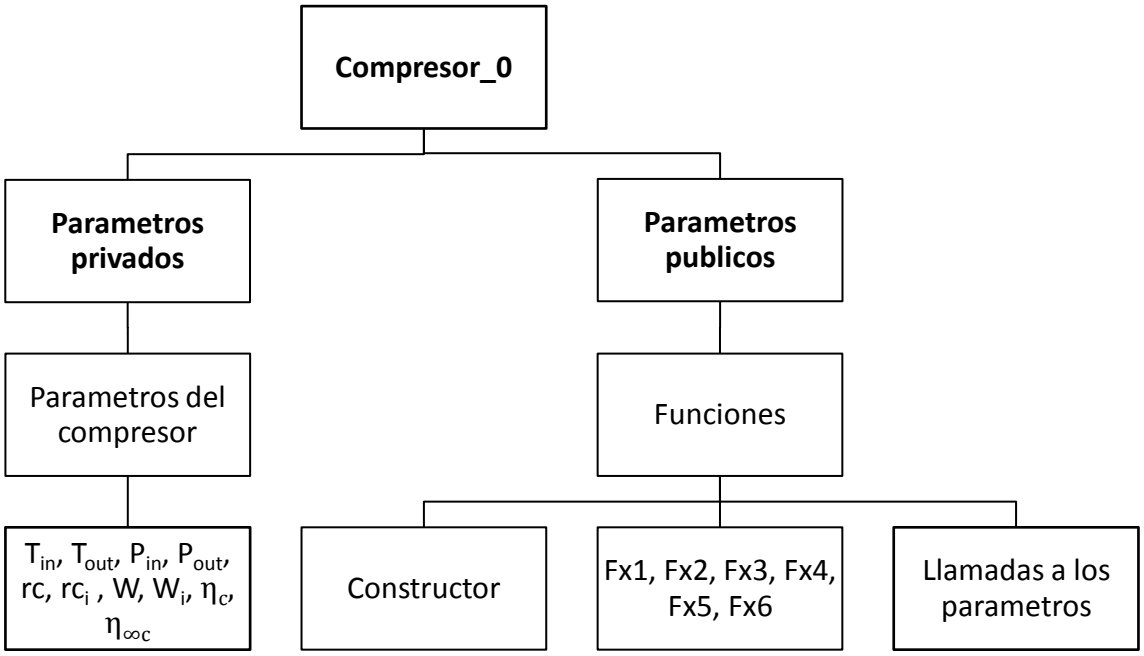


Fig. 2.7 Esquema del objeto compresor 1.

2.4. Resultados para la turbomaquinária ideal

Con el propósito de probar los elementos programados se crea un programa principal destinado a ejecutarlos de manera individual. A este programa se le llamará banco de pruebas, puesto que se intenta emular el funcionamiento de test y comprobaciones al cual es sometida una turbina de gas física en un laboratorio.

Por tanto, en el banco de pruebas se nos darán las opciones que podemos introducir, como por ejemplo asignar parámetros conocidos o desconocidos, guardar valores para posteriormente graficarlos y si fuera necesario cálculos inmediatos entre los parámetros existentes.

Por tanto, a partir de ahora cuando hagamos referencia a los bancos de prueba tendremos en cuenta una primera limitación matemática debido al método de Newton. Esta limitación establece que el sistema ha de tener tantas incógnitas como funciones existan en total. Una vez definida esta primera condición, en los apartados de resultados se vera como hay que introducir estos parámetros para obtener resultados lógicos, igualmente también se verá algunos ejemplos de los realizados para la creación de este trabajo.

Resultados del compresor ideal:

Para éste primer componente se comprobará que los resultados son lógicos con el programa anterior de una sola variable. Hay que tener en cuenta que de los parámetros que se dan a escoger en el banco de pruebas seleccionaremos 3 como incógnita y 3 como constantes. Por tanto, si cambiamos las unidades de la figura 2.6 al sistema internacional se tendría que ver como los valores se corresponden. Para comprobarlo se abren ambos programas y se introducen diferentes valores para los parámetros de la primera función en el programa de una sola variable y posteriormente se introduce cualquiera de las soluciones en el segundo programa, se vera que los resultados del segundo se corresponden enteramente al anterior.

Otra manera de comprobar el módulo es introduciendo los parámetros de otra manera distinta en la cual se intercambien las incógnitas por las constantes, como se muestra en la figura 2.9. En este caso se esperarían valores completamente iguales puesto lo único que se está realizando es el proceso inverso, pero resultará que habrá una cierta variación. Por ejemplo, si nos fijamos en la figura 2.9 se puede ver que a pesar que se han introducido los resultados de la temperatura, la relación de compresión y el trabajo íntegramente iguales que en la solución dada en el ejemplo de la figura 2.8 el valor de la temperatura no es el esperado de 100000 Pa. Este error será debido al número de decimales establecido para determinar que las funciones tienden a cero cuando se hacen las iteraciones para comprobar la convergencia de los resultados. En el caso de estos ejemplos dicho valor de resolución es de10^{-4}.

Fig. 2.8 Ejemplo de ejecución de la modelización 0.

Fig. 2.9 Ejemplo 2 de ejecución de la modelización 0.

Resultados de la turbina ideal:

En el caso de la turbina se podrían comprobar los resultados haciendo la inversa del ejemplo del compresor de la figura 2.8, es decir, pasaremos como constantes la temperatura y la presión a la entrada, 427.966355 y 400000 Pa respectivamente, así como la inversa de la relación de compresión, ¼=0.25. Como resultado de ésto obtenemos la figura 2.10. Como se puede apreciar, el valor del trabajo de la turbina tiene signo opuesto al del compresor, puesto que mientras que la turbina genera trabajo el compresor lo consume trabajo. A pesar de este cambio de signo vemos que el valor del trabajo es prácticamente el mismo, nuevamente debido a la resolución a la hora de evaluar las funciones para el método de Newton.

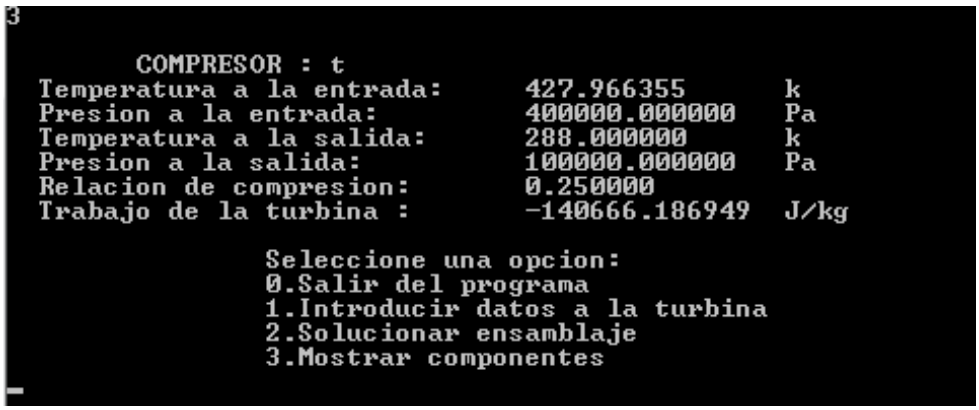


Fig. 2.10 Ejemplo 3 de ejecución de la modelización 0.

CAPÍTULO 3. MODELIZACIÓN 1: TURBOMAQUINARIA NO IDEAL

Debido a las fricciones de la maquinária entre sí y del aire con ésta hace que el ciclo de Bryton no sea ideal y que por tanto de este modo aparezcan perdidas. Como resultado de estas perdidas el ciclo se modificará, tal y como se puede apreciar en la figura 3.1. Dichas perdidas acabaran finalmente traduciéndose en un rendimiento que afectara a los trabajos ideales de los componentes.

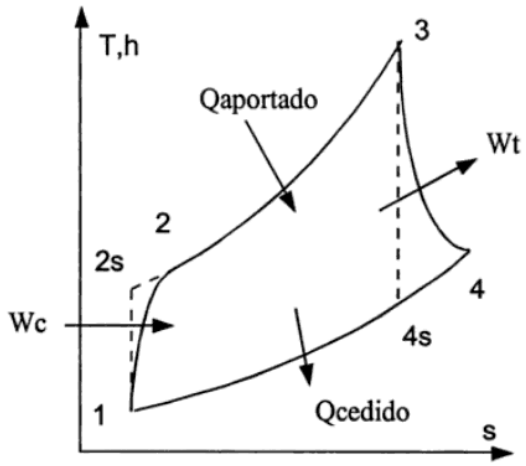


Fig. 3.1 Diagrama T-s del ciclo de Brayton no ideal. Gráfica extraída de la referencia [6]

3.1. Tipos de pérdidas

En el siguiente apartado se describirán algunos tipos de pérdidas que se pueden dar en los componentes que forman la turbomaquinária de una turbina de gas.

3.1.1. Pérdidas mecánicas

En todas las turbinas de gas, la potencia necesaria para mover al compresor se transmite directamente desde la turbina, sin ningún tipo de engranaje intermedio. Por lo tanto, cualquier pérdida que se produzca será debida únicamente al rozamiento en los cojinetes y a la ventilación. Dicha pérdida es muy pequeña, adquiriendo normalmente un valor de alrededor del 1% de la potencia necesaria para mover al compresor. Si llamamos η_m, definiéndola normalmente con el valor del 99%, el rendimiento de la transmisión, el trabajo requerido para esta función valdrá:

$$W = \frac{1}{\eta_m}\, Cp\, (T_f - T_o) \qquad \textbf{(3.1)}$$

La potencia empleada para accionar cualquier elemento auxiliar, como bombas de combustible o aceite, puede tenerse en cuenta frecuentemente con solo restarla de la potencia neta de la máquina. De forma que se puede considerar la potencia absorbida por la transmisión entre la turbina de gas y la carga.

3.1.2. Rendimiento isentrópico (Pérdidas isentrópicas)

El rendimiento de cualquier máquina cuya misión sea absorber o producir trabajo se suele expresar en forma de cociente entre el trabajo real y el ideal. Como las turbomáquinas son esencialmente adiabáticas, el proceso ideal será isentrópico, por lo que este rendimiento estará definido por las formulas (3.2) y (3.3), la primera para el compresor y la segunda para la turbina.

$$\eta_c = \frac{Wideal}{Wreal} = \frac{Cp*(T_{out\,ideal} - T_{in})}{Cp*(T_{out\,real} - T_{in})} \qquad \textbf{(3.2)}$$

$$\eta_t = \frac{Wreal}{Wideal} = \frac{Cp*(T_{out\,real} - T_{in})}{Cp*(T_{out\,ideal} - T_{in})} \qquad \textbf{(3.3)}$$

Como se puede ver en dichas formulas este rendimiento representa una variación en temperatura a la salida del componente, esto se puede ver claramente en la figura 3.2, como ejemplo en un compresor.

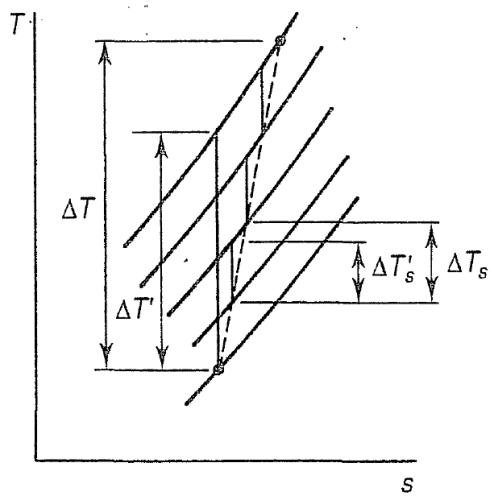


Fig. 3.2 Diagrama T-s del proceso isentrópico en un compresor con perdidas. Gráfico extraído de la referencia [1].

3.1.3. Rendimiento politrópico (Pérdidas politrópicas)

En el caso que se considerará un compresor de flujo axial consistente en una serie de escalonamientos sucesivos. Si a lo largo de todos ellos se adopta un diseño similar de álabes, es razonable admitir que el rendimiento isentrópico de un escalonamiento, η_s, se mantendrá invariable a lo largo del compresor. Pero como la distancia vertical entre dos líneas de presión de diagrama T-s se hace mayor a medida que aumenta la entropía acabara resultando $\eta_c < \eta_s$. Esto es debido a que el incremento de temperatura que tiene lugar en un escalonamiento a causa de la fricción hace que en el escalonamiento siguiente se requiera un trabajo mayor.

Por tanto, el rendimiento politrópico η_∞ se define como el rendimiento isentrópico de un escalonamiento elemental tal que se mantiene constante a lo largo de todo el proceso. Si se trata de una compresión, la definición anterior se puede expresar de manera matemática de la siguiente manera:

$$\eta_{\infty c} = \frac{dT'}{dT} = \text{constante} \qquad \textbf{(3.4)}$$

$$\frac{dT}{T} = \frac{(\gamma-1)}{\gamma}\frac{dP}{P} \qquad \textbf{(3.5)}$$

Añadiendo la formula (3.5), que es la forma diferencial de $T/P^{(\gamma-1)/\gamma}$ = constante, se pude despejar dT' y substituir en la formula (3.4) quedando de este modo:

$$\eta_{\infty c}\frac{dT'}{T} = \frac{(\gamma-1)}{\gamma}\frac{dP}{P} \qquad \textbf{(3.6)}$$

Si finalmente se integra entre la entrada y la salida y al ser η_∞ constante por definición se obtiene:

$$\eta_{\infty c} = \frac{\ln(P_{out}/P_{in})^{(\gamma-1)/\gamma}}{\ln(T_{out}/T_{in})} \qquad \textbf{(3.7)}$$

Desarrollando la formula (3.7) también se puede escribir del siguiente modo:

$$\frac{T_{out}}{T_{in}} = \left(\frac{P_{out}}{P_{in}}\right)^{(\gamma-1)/\gamma\eta_{\infty c}} \qquad \textbf{(3.8)}$$

Análogamente para la turbina, $\eta_{\infty t}$ = dT/dT', con lo que se demuestra que en una expansión entre la entrada y la salida:

$$\frac{T_{in}}{T_{out}} = \left(\frac{P_{in}}{P_{out}}\right)^{\eta_{\infty t}(\gamma-1)/\gamma} \quad \textbf{(3.9)}$$

Si se observan las formulas anteriores (3.6) y (3.7) se puede ver que si el termino del rendimiento politrópico es unitario dichas formulas son las de un proceso politrópico, y por tanto que un proceso no isentrópico es politrópico.

Existe una relación directa entre el rendimiento isentrópico y el rendimiento politrópico el cual se puede expresar como la formulas (3.10) y (3.11). A partir de estas ecuaciones y estableciendo un γ=1.4 se puede dibujar la figura 3.3, en la que se puede ver como varia de manera grafica η_c y η_t con la relación de compresión para un rendimiento politrópico fijo prestablecido del 85% en ambos casos.

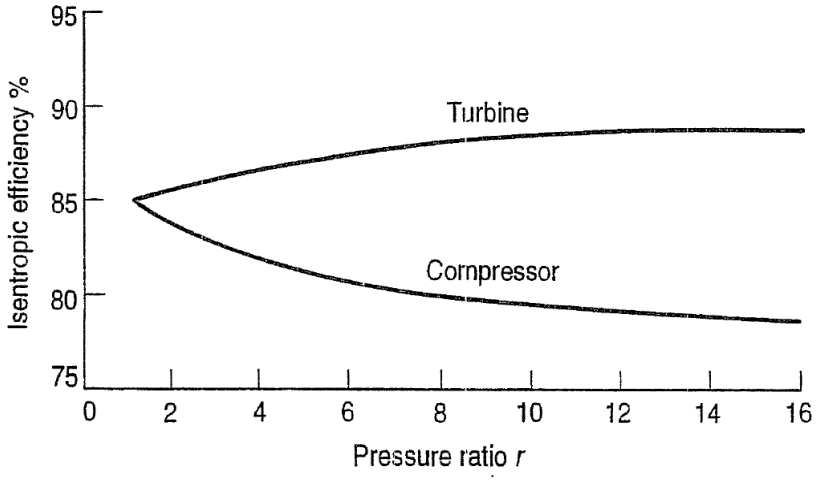


Fig. 3.3 Variación del rendimiento isentrópico del compresor y de la turbina con la relación de compresión para un rendimiento politrópico del 85%. Gráfico extraído de la referencia [1].

$$\eta_c = \frac{(P_{out}/P_{in})^{(\gamma-1)/\gamma} - 1}{(P_{out}/P_{in})^{(\gamma-1)/\gamma}\eta_{\infty c} - 1} \quad \textbf{(3.10)}$$

$$\eta_t = \frac{1-\left(\frac{1}{P_{in}/P_{out}}\right)^{\eta_{\infty t}(\gamma-1)/\gamma}}{1-\left(\frac{1}{P_{in}/P_{out}}\right)^{(\gamma-1)/\gamma}} \quad \textbf{(3.11)}$$

3.2. Programación de la turbomaquinaria no ideal

En este caso para programar los componentes usaremos la teoría del primer capitulo para turbomaquinaria ideal y se le añadirá los conceptos de rendimiento politrópico e isentrópico. Así como se mejoraran ciertas funciones con el propósito de que optimicen y hagan más robusto el programa.

3.2.1. Compresor

Para el compresor se usaran las formulas (3.2) y (3.8). Se usaran también la (2.7) y (2.8) del capítulo anterior, y la (2.3) intercambiando la temperatura genérica a la salida por la ideal. Por tanto, resultaran las siguientes funciones:

$$fx_1 = \frac{T_{out\ ideal}}{T_{in}} - \left(\frac{P_{out}}{P_{in}}\right)^{(\gamma-1)/\gamma} \quad \textbf{(3.12)}$$

$$fx_2 = r_c - \frac{P_{out}}{P_{in}} \quad \textbf{(3.13)}$$

$$fx_3 = 1 - \frac{Cp*(T_{out\ real} - T_{in})}{Wreal} \quad \textbf{(3.14)}$$

$$fx_4 = 1 - \frac{Cp*(T_{out\ ideal} - T_{in})}{Wideal} \quad \textbf{(3.15)}$$

$$fx_5 = \eta_c - \frac{Wideal}{Wreal} \quad \textbf{(3.16)}$$

$$fx_6 = \frac{T_{out\ real}}{T_{in}} - \left(\frac{P_{out}}{P_{in}}\right)^{(\gamma-1)/\gamma\eta_{\infty c}} \quad \textbf{(3.17)}$$

Como se puede observar estas formulas se han intentado normalizar a valores los más unitarios posibles, con ello se ganara precisión en los resultados del programa.

Por tanto, cuando se programe el modulo se tendrá el objeto definido según el esquema de la figura 2.7 del capitulo anterior, añadiendo los parámetros del compresor añadidos en este capitulo y las nuevas funciones.

3.2.2. Turbina

En el caso de la turbina se emplearan, de manera análoga, las funciones (3.3), (3.9), (2.7) y (2.8), mientras que las formulas que contienen las eficiencias se verán modificadas tal y como se ha visto en los respectivos apartados al início de este capítulo, obteniendo así las fórmulas (3.18) y (3.19).

$$fx_5 = \eta_c - \frac{Wreal}{Wideal} \quad \textbf{(3.18)}$$

$$fx_6 = \frac{T_{out\,real}}{T_{in}} - \left(\frac{P_{out}}{P_{in}}\right)^{\eta_{\infty c}(\gamma-1)/\gamma} \quad \textbf{(3.19)}$$

Igualmente como en el apartado anterior las fórmulas estarán normalizadas. Una vez se tienen las fórmulas el objeto turbina se puede describir con el mismo esquema que el compresor, que en se encuentra en la figura 3.4.

3.2.3. Mejoras de algunas funciones con BLAS y LAPACK

En éste grado de modelización se intentará mejorar el funcionamiento del programa del banco de prueba, introduciendo de las librerías BLAS y LAPAK ciertas rutinas que permitirán obtener una resolución más eficiente de las operaciones vectoriales y matriciales necesarias en la implementación del método de Newton-Raphson multivariable. Dichas rutinas son:

- **daxpy:** Esta primera rutina se encarga de realizar la resta de vectores que sean necesarias. Por ello la usaremos para hacer la resta entre $f(x_n)$ y $f(\Delta x_n)$ y para restar a x_n el resultado del término $f(x_n) / f'(x_n)$. La primera se encuentra en los programas principales o bancos de pruebas, la segunda se hallará dentro del código del módulo de método numérico.

- **dscal:** Esta sirve para multiplicar un escalar por un vector por tanto en nuestro caso la usaremos para implementar la división realizada para obtener las aproximaciones de las derivadas, consecuentemente el valor de dicho escalar será $1/ \Delta x$.

- **dnrm2:** En estas librerías podemos encontrar la norma 2, que no es más que la raíz de las suma de los cuadrados de los números que contiene el vector introducido por referencia. En éste trabajo esto permitirá saber si las funciones convergen o no, como también establecer de una manera más eficiente una condición para el “while” donde se itera el método de Newton.

3.3. Resultados para la turbomaquinária no ideal

En este apartado se verá como para testear este grado de modelización se usaran ejemplos de la referencia [1] con tal de probar que los valores calculados son correctos. Posteriormente se crearan gráficos propios con tal de ver el comportamiento y confirmar conceptos teóricos.

Compresor no ideal

La primera comprobación de este componente se hará con el ejemplo que encontraremos en la referencia [1] en la pagina 58, en este caso solo nos fijaremos en los datos concernientes al compresor. Por tanto a partir de este ejemplo se le podrán introducir al programa la temperatura de entrada, la presión de entrada, la relación de compresión y la eficiencia isentrópica, como se puede apreciar en la figura 3.5. Como resultado de esta comprobación veremos que los valores de salida son iguales a los de la solución del problema e incluso el programa ofrece mayor numero de decimales. Solo se tendría que remarcar que el trabajo del compresor del programa hay que dividirlo por 0.99 para equipararlo al trabajo que proporciona el ejemplo el cual incluye un rendimiento mecánico de la transmisión.

```
                Seleccione una opcion:
                0.Salir del programa
                1.Introducir datos al compresor
                2.Solucionar ensamblaje
                3.Mostrar componentes
1

        ID. COMPRESOR: c

Determine el valor de las variables conocidas y asigne un -1 a las desconocidas:

Temperatura a la salida(k):-1
Presion a la salida(Pa):-1
Temperatura a la entrada(k):288
Presion a la entrada(Pa):100000
Relacion de compresion:4
Eficiencia politropica:-1
Eficiencia isentropica:0.85
Trabajo del compresor(J/kg):-1

                Seleccione una opcion:
                0.Salir del programa
                1.Introducir datos al compresor
                2.Solucionar ensamblaje
                3.Mostrar componentes
2

                Seleccione una opcion:
                0.Salir del programa
                1.Introducir datos al compresor
                2.Solucionar ensamblaje
                3.Mostrar componentes
3

        COMPRESOR : c
  Temperatura a la entrada:     288.000000      k
  Presion a la entrada:         100000.000000   Pa
  Temperatura a la salida:      452.666300      k
  Presion a la salida:          400000.000000   Pa
  Relacion de compresion:       4.000000
  Eficiencia politropica:       0.875915
  Eficiencia isentropica:       0.850000
  Trabajo del compresor :       165489.631821   J/kg
```

Fig. 3.5 Ejemplo 1 de ejecución de la modelización 1.

Con ésta prueba no se determina si los valores de las eficiencias están relacionados de manera correcta. Por tanto se realizará un gráfico fijando todos los parámetros a la entrada anteriores constantes y en vez de definir el rendimiento isentrópico se marcará el rendimiento politrópico como una constante de 0.85, y una vez fijado estos parámetros, se variara la relación de compresión. De este modo el gráfico resultante se podrá comparar con el teórico de la figura 3.6. Por tanto, como se ve en la figura 3.6 los valores del compresor coinciden.

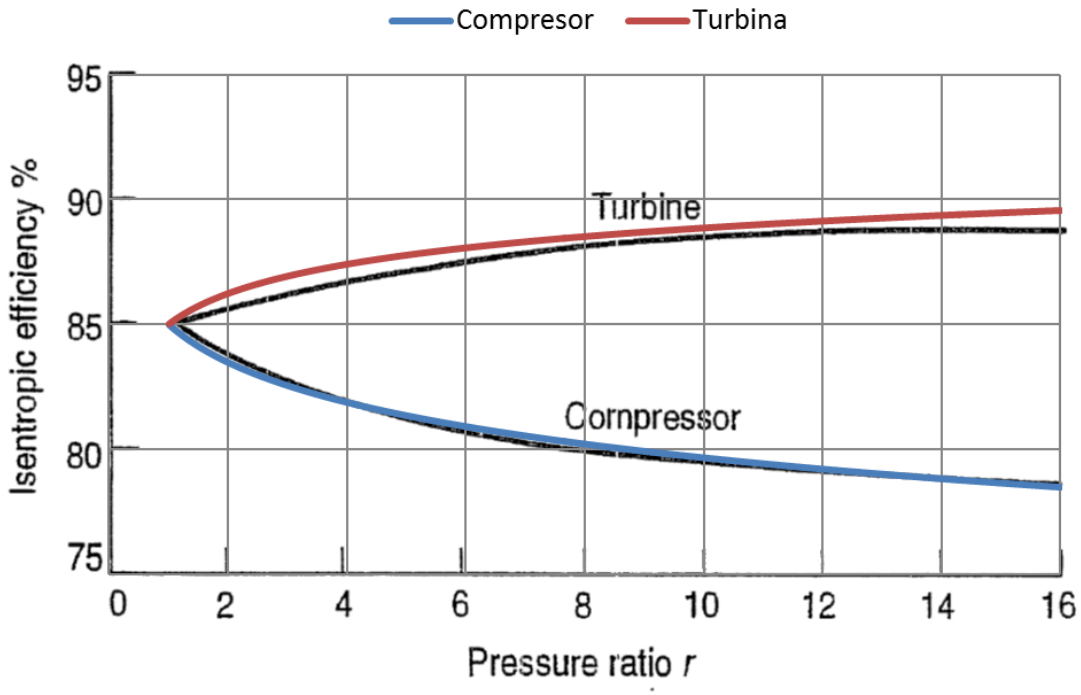


Fig. 3.6 Variación del rendimiento isentrópico del compresor y de la turbina con la relación de compresión para un rendimiento politrópico del 85%.

Turbina no ideal

Para la turbina también se comprobaran los resultados del ejemplo usado en el compresor de esta modelización, correspondiente a la referencia [1]. Solo hay que puntualizar que como se trata de un ejemplo de motor completo habrá que buscar los valores que correspondan a los de la turbina, puesto que el enunciado del ejercicio no los da específicamente. Como se ve en la figura 3.7 se puede comprobar que los resultados como en el caso del compresor dan valores iguales a los del ejemplo salvando la diferencia de decimales significativos.

Si se realiza la gráfica de comparación de rendimientos se verá que al compararla con la teórica los resultados de la turbina tienen cierta desviación, posiblemente porque el número de puntos escogidos para crear la gráfica teórica fuera menor, tal y como se puede ver en la figura 3.6.

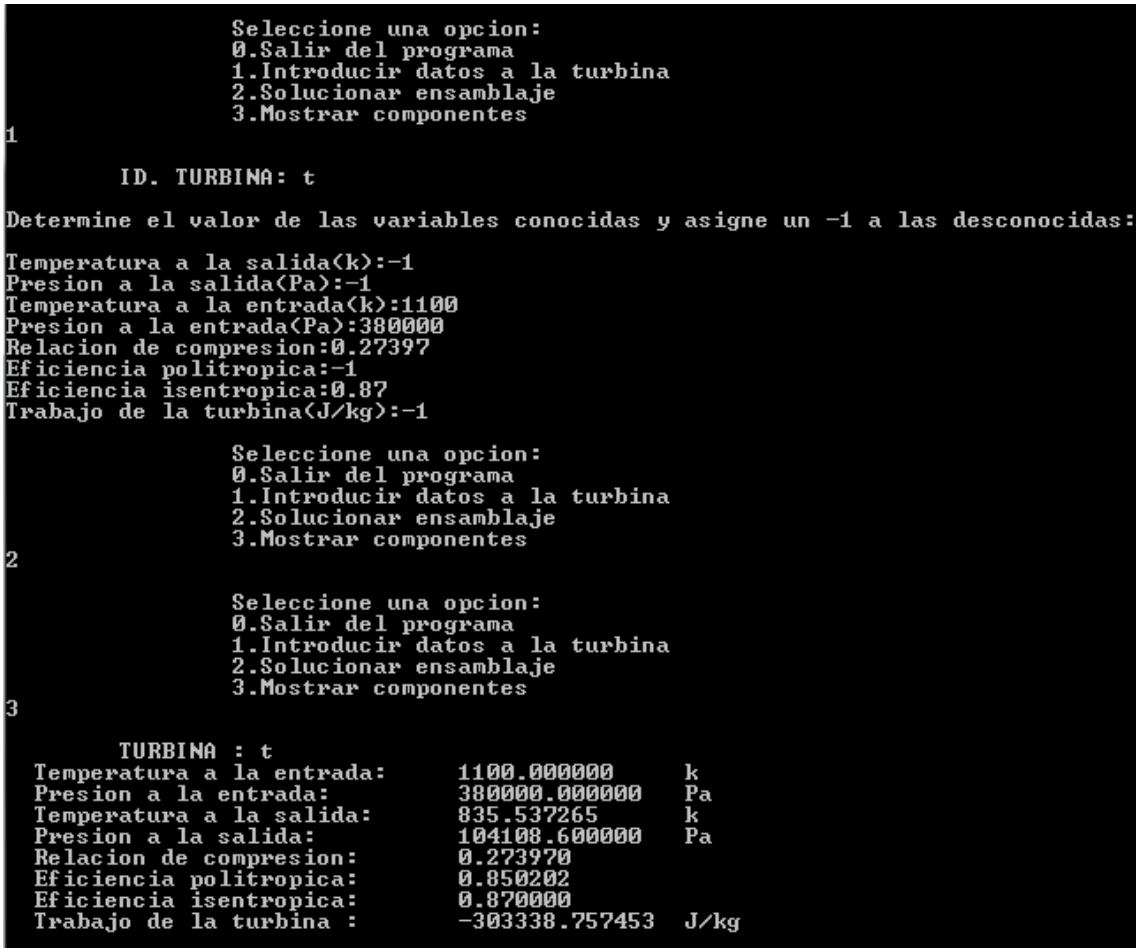


Fig. 3.7 Ejemplo 2 de ejecución de la modelización 1.

CAPÍTULO 4. MODELIZACIÓN 2: MAGNITUDES TOTALES

En este capítulo se modificaran los parámetros de entrada y salida de los diferentes componentes en función del flujo másico y la superficie de la sección de entrada o salida. Por este motivo también se empezará a hacer distinción entre compresores y fan (ventilador). Al hablar del fan también se hablará de la hélice ya que ambos están destinados a proporcionar un empuje adicional al resto del motor.

4.1. Magnitudes totales

Hasta el momento se estaba suponiendo que el flujo de aire que circulaba por los componentes tenía una velocidad nula. A partir de ahora, se supondrá que el flujo esta en movimiento y como consecuencia se tendrá una componente estática y una cinética. Esto es así, ya que entre los términos de la ecuación de energía para un flujo estacionario se puede definir la entalpía total h_0 como la entalpia que tendría una corriente de gas de entalpia h y velocidad C si se la frena hasta el reposo adiabáticamente y sin realizar trabajo.

$$Q = (h_0 - h) + \frac{1}{2}(0 - C^2) = 0 \qquad \textbf{(4.1)}$$

Por tanto, h_0 se reduce a:

$$h_0 = h + \frac{1}{2}C^2 \qquad \textbf{(4.2)}$$

Si se sigue desarrollando esta explicación veremos como para un gas perfecto se puede sustituir la entalpia (h) por el termino c_p*T, con lo que se halla la temperatura total, T_0, como se aprecia en la formula (4.3).

$$T_0 = T + {}^{C^2}/_{2C_p} \qquad \textbf{(4.3)}$$

Al termino $C^2/2c_p$ se le domina temperatura dinámica y T será la temperatura estática.

Por otra parte, si disminuye la velocidad y la temperatura aumenta, se produce un aumento de presión. Nuevamente se definirá una presión total P_0 de manera

análoga a T_0, como se puede ver en la formula (4.4); pero con la salvedad de que el gas no solo se supondrá que es frenado hasta el reposo adiabáticamente si no también reversiblemente, es decir, isotrópicamente. (No confundir la presión total con la presión de pitot).

$$\frac{P_0}{P} = \left(\frac{T_0}{T}\right)^{\gamma/(\gamma-1)} \quad \textbf{(4.4)}$$

Finalmente, para no dejarla dicha presión en función de la temperatura total se substituirá la formula (4.3), $c_p=\gamma R/(\gamma-1)$ y $P=\rho RT$ en la (4.4), obteniendo la siguiente ecuación:

$$P_0 = P\left(1 + \left(\frac{\rho * C^2}{2P}\,\frac{(\gamma-1)}{\gamma}\right)\right)^{\gamma/(\gamma-1)} \quad \textbf{(4.5)}$$

Los diferentes estados pueden señalarse en un diagrama T-s, tal como se ve en la figura 4.1, que representa una compresión entre los estados estáticos 1 y 2. Aclarar, que los símbolos que llevan notación prima hacen referencia a los estados ideales.

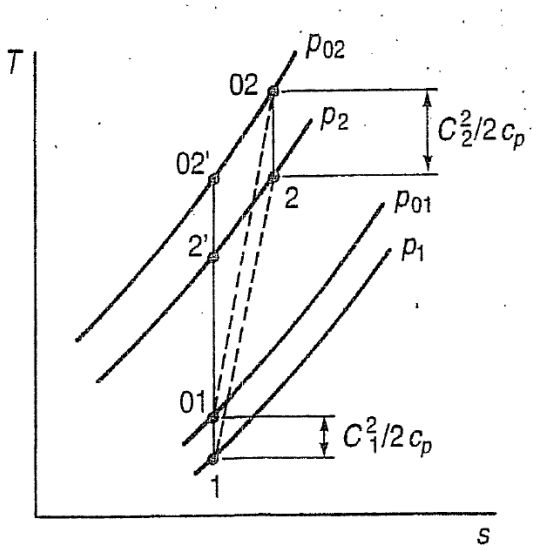


Fig. 4.1 Representación T-s de los estados totales en contraposición de los estáticos. Extraido de la referencia [1].

4.1.1. Flujo másico reducido

Una vez hemos introducido los componentes dinámicos, los cuales contienen la velocidad del fluido pasaremos estas velocidades de m/s a Mach con tal de

facilitar el posterior trabajo con áreas y flujos másicos. Las fórmulas resultantes de este paso serán:

$$T_0 = T\ (1 + \left(\frac{(\gamma-1)}{2}\ M^2\right)) \quad \textbf{(4.6)}$$

$$P_0 = P(1 + \left(\frac{(\gamma-1)}{2}\ M^2\right))^{\gamma/(\gamma-1)} \quad \textbf{(4.7)}$$

Mediante el mach que se ha obtenido de las fórmulas anteriores, se podrá relacionar con un parámetro adimensional conocido como flujo másico reducido, ξ, con el área de la sección y el flujo másico, en kg/s, que pasa por dicha sección:

$$\xi = M\ \sqrt{\gamma/R}\ (1 + \left(\frac{(\gamma-1)}{2}\ M^2\right))^{-(\gamma+1)/2(\gamma-1)} \quad \textbf{(4.8)}$$

$$\xi = \frac{\dot{m}\ \sqrt{T}}{P\ A} \quad \textbf{(4.9)}$$

Por tanto, se puede ver que el motor se podría escalar según interesase a posterior, una vez se tiene el flujo másico reducido.

4.2. Fan

El fan es parte de un tipo de motor de turbina concreto usado para la aviación, conocido como turbofán. Este tipo de motor fue ideado con el propósito de mejorar el rendimiento propulsivo del motor de reacción al disminuir la velocidad media del chorro, especialmente a velocidades subsónicas elevadas. Como consecuencia añadida también se consiguió una reducción de la velocidad media del chorro también comportaba una reducción en el ruido que este produce. El funcionamiento de éste es muy similar a del motor a reacción la única diferencia es que después del difusor, si se trata de una configuración habitual, la corriente de entrada se encuentra con el ventilador y en este se separa el flujo entre el resto del motor (compresores, cámara de combustión, turbina y tobera) y una tobera separada, tal como se ve en la figura 4.2. Así pues, el empuje también contara de dos componentes, el empuje de corriente fría y el empuje de corriente caliente. Comentar que en la figura las corrientes son separadas, pero existe también la posibilidad de mezclar dichas corrientes al final como un único chorro de menor velocidad.

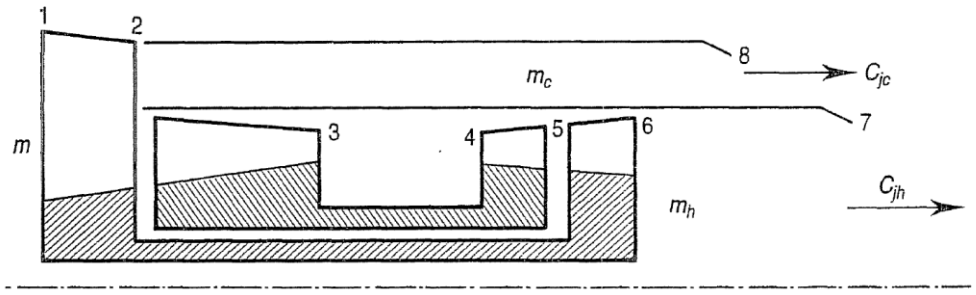


Fig. 4.2 Representación esquemática de un turbofán de doble eje. Extraído de la referencia [1].

Por tanto para poder evaluar correctamente el turbofán se hace necesario definir los parámetros que nos introduce su componente característico, el fan. Este nos introduce un nuevo concepto conocido como relación de by-pass, también conocido como grado de derivación, es cual se define como el cociente entre los gastos del conducto de corriente fría y del corriente caliente como muestra en la formula (4.10). Por tanto, con esta formula y sabiendo que el flujo másico total a la entrada es la suma del flujo de corriente fría y caliente (formula (4.11)). Se nos permite dimensionar el fan en dos secciones o, de manera teórica e idealizada, en dos compresores en paralelo.

$$B = \frac{\dot{m}_{\,corriente\,fira}}{\dot{m}_{\,corriente\,caliente}} \qquad \textbf{(4.10)}$$

$$\dot{m} = \dot{m}_{\,corriente\,fira} + \dot{m}_{\,corriente\,caliente} \qquad \textbf{(4.11)}$$

4.3. Hélice

La finalidad de la hélice es extraer potencia mecánica de la turbina y transformar parte de ella en forma de empuje del chorro de aire. En este caso, por tanto, es necesario combinar potencia mecánica y empuje, lo que puede hacerse de diversas maneras, pero interviniendo en todas ellas el conocimiento de la velocidad de vuelo. De este modo queda claro que la hélice es un elemento propulsivo, que forma parte de un tipo de motor de turbina conocido como turbohélice.

Este tipo de motor es fácilmente comparable con uno de émbolo en cuyo caso la única diferencia es que la hélice no está ligada a un motor a reacción si no a uno de pistones. Por tanto en términos generales se puede definir la potencia de empuje (TP) de la hélice como función de una potencia mecánica (SP), un rendimiento de hélice (η_{pr}) y un empuje del flujo (F), resultando la siguiente formula:

$$TP = (SP)\eta_{pr} + F\ C_{\ a} \quad \textbf{(4.12)}$$

En la práctica, la potencia mecánica será responsable de una gran proporción de salto entálpico disponible a la salida del generador de gas, por lo que la potencia de empuje dependerá en gran medida del rendimiento de la hélice, que puede variar de forma significativa con las condiciones de vuelo.

El rendimiento combinado de la turbina de potencia, la hélice y el mecanismo de reducción necesario, es muy inferior al de una tobera propulsiva equivalente. Es decir, el rendimiento en un turbohélice es menor que en un turborreactor o un turbofán, solo el rendimiento será mejor a velocidades bajas donde el rendimiento propulsivo de la hélice será mejor que el proporcionado por el motor a reacción. Es por este motivo, y por su baja producción de ruido acústico, lo que hace que se use normalmente este tipo de motores para aviones que realizan trayectos relativamente cortos. Finalmente, se puede ver que el fan de paso variable no es más que una hélice rodeado de un conducto.

4.4. Programación de la modelización 2

En este caso no habrán diferencias conceptuales acusadas como se verá entre la programación de los componentes compresor y turbina de esta sección con los de la anterior puesto ambos son objetos sencillos y siguen el mismo esquema organizativo.

4.4.1. Compresor y turbina

Puesto que normalmente y lo más habitual es trabajar con parámetros totales, y además como se ha visto que se puede calcular directamente el flujo másico y las áreas de las secciones a la entrada y salida del componente con la formula (4.9), se puede ahorrar la inclusión de las formulas relacionadas con valores estáticos. Como consecuencia de ello las funciones que se definirán en la programación tanto de compresor como de turbina serán las del tema anterior y añadiremos las siguientes:

$$fx_7 = \xi_{in} - M_{in}\sqrt{\gamma/R}\ \left(1 + \left(\frac{(\gamma-1)}{2}\ M_{in}^2\right)\right)^{-(\gamma+1)/2(\gamma-1)} \quad \textbf{(4.13)}$$

$$fx_8 = \xi_{out} - M_{out}\sqrt{\gamma/R}\ \left(1 + \left(\frac{(\gamma-1)}{2}\ M_{out}^2\right)\right)^{-(\gamma+1)/2(\gamma-1)} \quad \textbf{(4.14)}$$

$$fx_9 = \xi_{in} - \frac{\dot{m}_{in}\sqrt{T_{in}}}{P_{in}\ A_{in}} \quad \textbf{(4.15)}$$

$$fx_{10} = \xi_{out} - \frac{\dot{m}_{out}\sqrt{T_{out}}}{P_{out}\,A_{out}} \quad \textbf{(4.16)}$$

$$fx_{11} = \dot{m}_{in} - \dot{m}_{out} \quad \textbf{(4.17)}$$

En ésta última función se está definiendo que el flujo másico en los componentes se mantendrán constantes y que por tanto se encontraran variaciones en el área de las secciones de este modo se podrá dimensionar cada componente.

4.4.2. Fan

El fan representará un pequeño cambio en la definición de los objetos hasta ahora puesto que éste contendrá a su vez una serie de objetos. Esto es debido a que se pretende ver el fan como un conjunto de dos compresores en paralelo. Por tanto, y de manera que se pueda comprender mejor, a continuación se puede ver una representación esquemática del objeto:

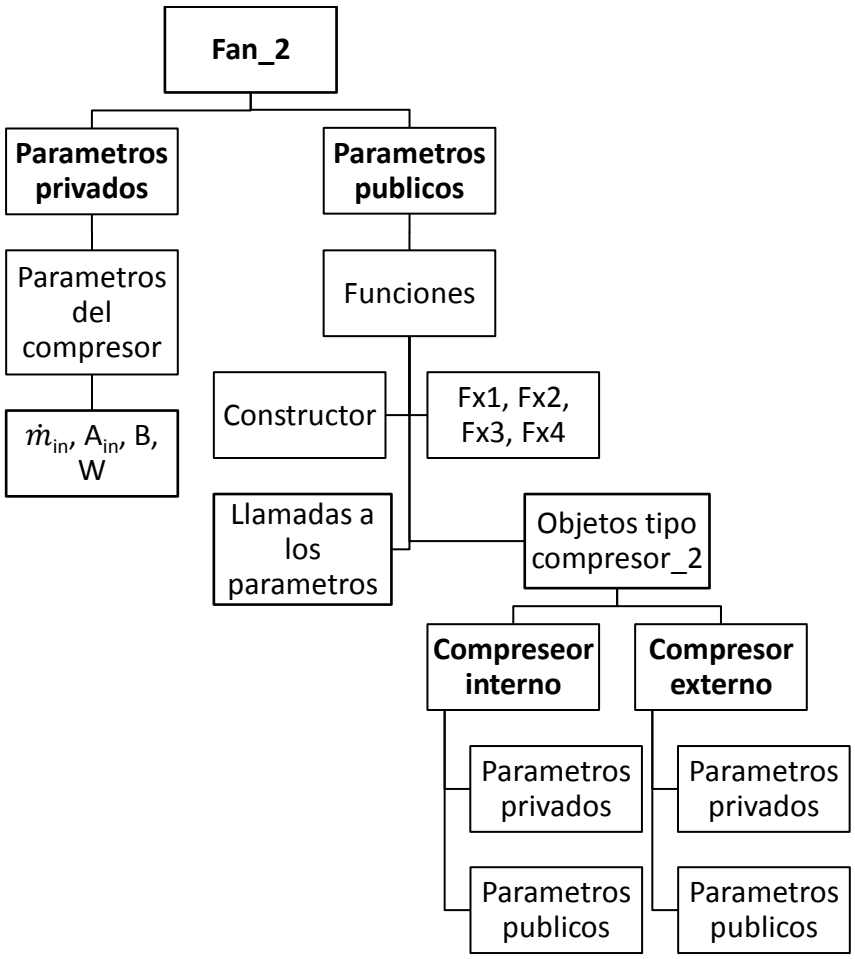


Fig. 4.3 Esquema del objeto compresor 1.

Obsérvese que los objetos incluidos dentro del objeto fan también contienen dentro parámetros privados y públicos los cuales de manera indirecta son al mismo tiempo parámetros privados y públicos del fan, de este modo solo hará falta llamarlos cuando sea necesario.

4.5. Resultados de la modelización 2

Para testear este grado de modelización se usaran ejemplos de la referencia [1], así como del programa *Gas Turb 11* en el cual se le pueden introducir parámetros similares al nuestro con tal de probar que los de salida se corresponden. Puesto que anteriormente ya se ha visto el como se compara con ejemplos de libro en este apartado se hará referencia solo al *Gas Turb 11*.

La manera de proceder con el *Gas Turb 11* será creando primeramente el ejemplo y posteriormente ejecutar nuestro programa, esto es debido a que de esta manera se pueden evitar cambios de valores debido a módulos previos o posteriores. Estos cambios son debidos a que en el *Gas turb 11* solo hay la posibilidad de marcar unos parámetros muy concretos, veremos ejemplos de este incidente en los apartados concretos de los componentes.

Compresor y turbina

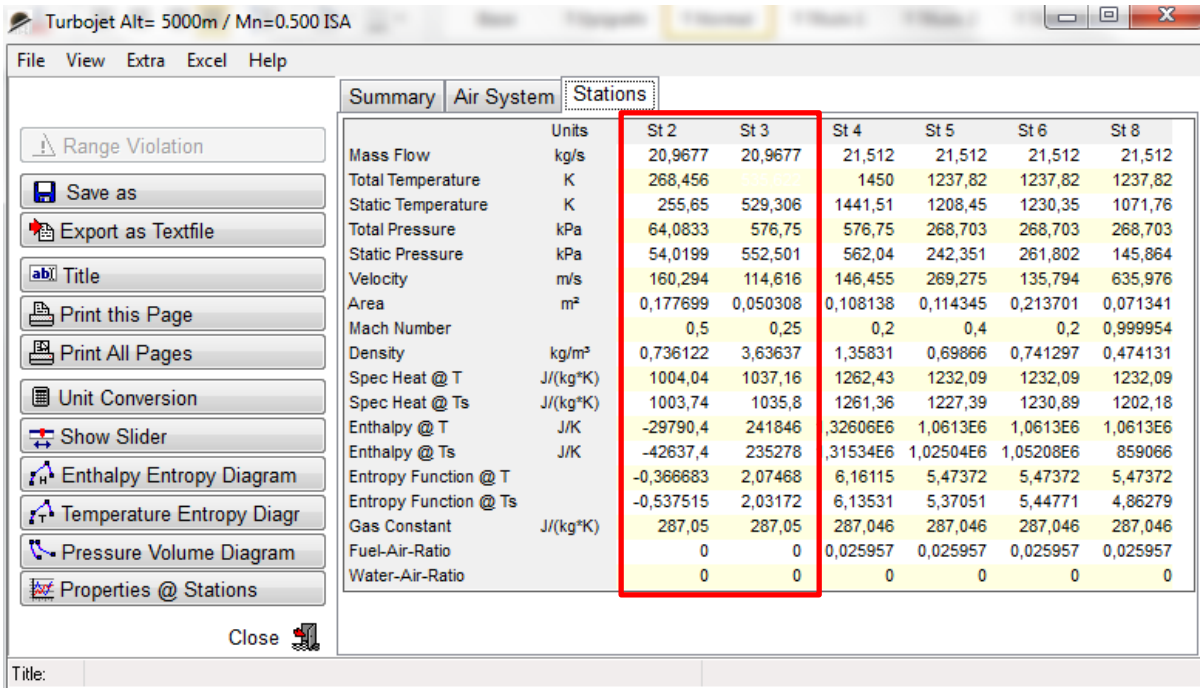
Para poder testear estos componentes se ha seleccionado en el *Gas Turb 11* el turbojet simple y se ha seleccionado en la versión más completa del programa. En este modo se puede visualizar una pestaña extra con los datos de las etapas intermedias del motor, que son los que interesan para poder hacer una buena comparación.

Explicaremos un ejemplo con el compresor, en el caso que se quisiera hacer con la turbina se seguirían los mismos pasos. Para empezar se introducen los datos genéricos en la ventana anterior asegurándose de que el mayor número de eficiencias tomen valor unitario, con el propósito de que no causen ningún efecto en el módulo que queremos testear. Una vez hecho esto seleccionaremos la pestaña de “Stations” y se escogerán la primera y segunda columna, estas se pueden ver marcadas en la figura 4.4.

De estas columnas interesará fijarse en los datos de temperaturas y presiones totales, las velocidades de mach, los flujos másicos y las áreas de las distintas secciones. Otros parámetros a tener en cuenta podrían ser la contante de los gases y la velocidad en m/s, y de este modo calcular también γ, para poder introducirlas en nuestro programa pero después de algunos test se vería que el efecto que tienen sobre los resultados es ínfimo.

Antes de introducir los valores al programa habría que cerciorarse que tiene sentido aquello que buscamos y que no se pudiera sobrecargar banco de pruebas, un ejemplo muy claro de esto es el hecho de no dar los flujos másicos tanto a entrada como a salida puesto ya existe una unión entre ellas. Otro

ejemplo no tan evidente podría ser el introducir en la entrada el mach, y también el flujo másico y área de entrada puesto que el flujo másico reducido se vería en conflicto. Una vez identificados los parámetros que nos interesan se introducirán como se muestra en la figura 4.5.



	Units	St 2	St 3	St 4	St 5	St 6	St 8
Mass Flow	kg/s	20,9677	20,9677	21,512	21,512	21,512	21,512
Total Temperature	K	268,456		1450	1237,82	1237,82	1237,82
Static Temperature	K	255,65	529,306	1441,51	1208,45	1230,35	1071,76
Total Pressure	kPa	64,0833	576,75	576,75	268,703	268,703	268,703
Static Pressure	kPa	54,0199	552,501	562,04	242,351	261,802	145,864
Velocity	m/s	160,294	114,616	146,455	269,275	135,794	635,976
Area	m²	0,177699	0,050308	0,108138	0,114345	0,213701	0,071341
Mach Number		0,5	0,25	0,2	0,4	0,2	0,999954
Density	kg/m³	0,736122	3,63637	1,35831	0,69866	0,741297	0,474131
Spec Heat @ T	J/(kg*K)	1004,04	1037,16	1262,43	1232,09	1232,09	1232,09
Spec Heat @ Ts	J/(kg*K)	1003,74	1035,8	1261,36	1227,39	1230,89	1202,18
Enthalpy @ T	J/K	-29790,4	241846	,32606E6	1,0613E6	1,0613E6	1,0613E6
Enthalpy @ Ts	J/K	-42637,4	235278	,31534E6	1,02504E6	1,05208E6	859066
Entropy Function @ T		-0,366683	2,07468	6,16115	5,47372	5,47372	5,47372
Entropy Function @ Ts		-0,537515	2,03172	6,13531	5,37051	5,44771	4,86279
Gas Constant	J/(kg*K)	287,05	287,05	287,046	287,046	287,046	287,046
Fuel-Air-Ratio		0	0	0,025957	0,025957	0,025957	0,025957
Water-Air-Ratio		0	0	0	0	0	0

Fig. 4.4 Tabla con los valores de los estados de un turbojet, modelizado por el programa Gas Turb 11.

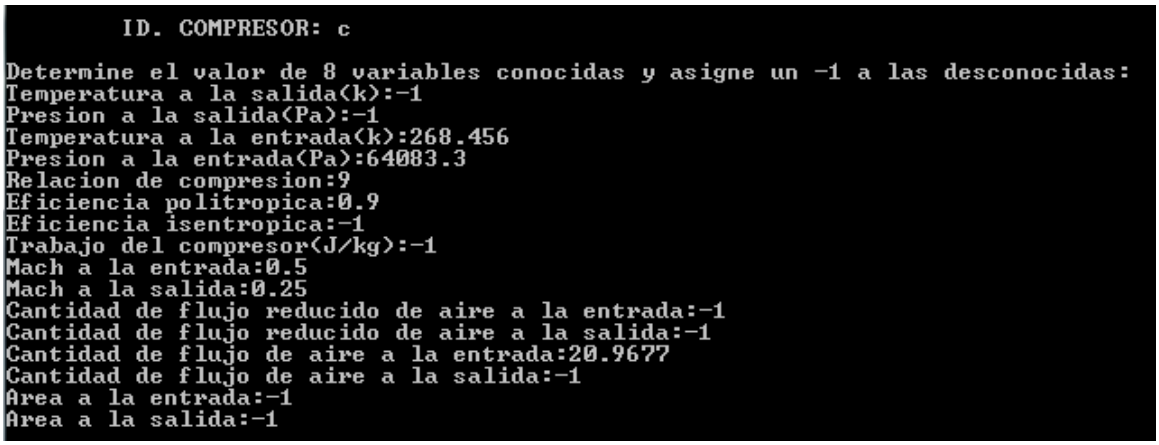


Fig. 4.5 Datos de entrada para un ejemplo de compresor 2.

Una vez se obtienen los resultados de la figura 4.6, se puede verificar como los datos de salida son iguales con un cierto margen de error. Este error puede venir dado por ejemplo a que el método numérico usado en el *Gas Turb 11* sea diferente, o porque se usan resoluciones diferentes, o bien porque el *Gas Turb*

11 al trabajar con motores completos se haya introducido algún parámetro que influyera de algún modo en estos resultados.

```
      COMPRESOR : c
Temperatura a la entrada:                          268.456000      k
Presion a la entrada:                              64083.300000    Pa
Mach a la entrada:                                 0.500000
Cantidad de flujo reducido de aire a la entrada:   0.030164
Cantidad de flujo a la entrada:                    20.967700       Kg/s
Area de la seccion de entrada:                     0.177728        m^2

Temperatura a la salida:                           539.271222      k
Presion a la salida:                               576749.700000   Pa
Mach a la salida:                                  0.250000
Cantidad de flujo reducido de aire a la salida:    0.016821
Canditadad de flujo a la salida:                   20.967700       Kg/s
Area de la seccion de salida:                      0.050191        m^2

Relacion de compresion:                            9.000000
Eficiencia politropica:                            0.900000
Eficiencia isentropica:                            0.865835
Trabajo del compresor :                            272169.298028   J/kg
```

Fig. 4.6 Resultados para un ejemplo de compresor 2.

Para comprobar conceptos teóricos a continuación se presenta una gráfica en la cual se ha comparado el Mach de salida de cada componente con sus respectivas superficies de salida. Para poder elaborar dicha gráfica se han introducido valores iniciales idénticos, temperatura de entrada de 288K, presión de entrada 100000 Pa y flujo másico de 100 Kg/s. Por otra parte, se han fijado los mismos rendimientos politrópicos y las relaciones de compresión. Dando todos estos datos, se obtiene una sección de entrada constante de 0.562612 m^2 y lógicamente se pueden ver como las secciónes son inferiores a la salida del compresor que a la entrada de la turbina.

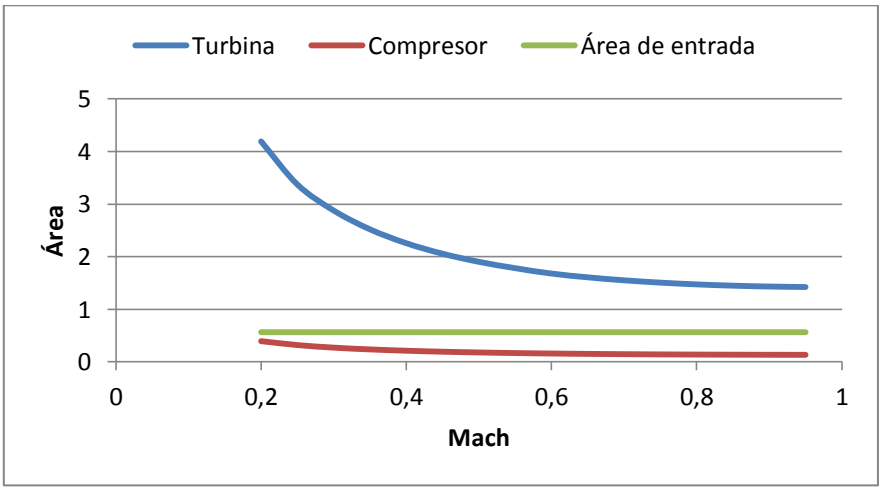


Fig. 4.7 Gráfica comparativa de las superficies en la sección de salida.

De este modo, podemos comprobar que se puede controlar el mach de salida de cada componente en función de las dimensiones de salida de éste.

Fan

Para el caso del fan usaremos el mismo método que antes, salvo porque se selecciona como tipo de motor un turbofan de doble eje sin mixer, es decir, el cual no mezcla a la salida los chorros de corriente fría con los de corriente caliente. Se ha de aclarar que las etapas marcadas en la figura 4.8 corresponden al fan de la siguiente manera: la etapa St2 el la entrada al fan, la St21 es la salida interior del fan, es decir, la entrada al compresor de alta; y por último, la etapa St13 es la salida de la parte exterior del fan.

Summary | Air System | Stations

	Units	St 2	St 21	St 25	St 3	St 4	St 44	St 45	St 5	St 6	St 8	St 13
Mass Flow	kg/s	42,5241	10,631	10,631	10,631	10,909	10,909	10,909	10,909	10,909	10,909	31,893
Total Temperature	K	244,442	286,625	286,625	528,227	1450	1254,68	1254,68	1119,96	1119,96	1119,96	286,625
Static Temperature	K	232,78	277,724	272,955	524,221	1431,03	1208,79	1208,79	1077,86	1109,23	966,587	272,955
Total Pressure	kPa	34,509	56,9399	56,9399	398,579	398,579	198,039	198,039	115,611	115,611	115,611	56,9399
Static Pressure	kPa	29,0887	50,9951	47,9996	387,744	376,145	168,725	168,725	98,4033	110,987	62,6034	47,9996
Velocity	m/s	152,973	133,642	165,616	91,27	218,905	336,623	336,623	318,839	161,593	605,731	165,616
Area	m²	0,638553	0,124358	0,104781	0,045204	0,054422	0,066644	0,066644	0,107577	0,193671	0,079818	0,314344
Mach Number		0,5	0,4	0,5	0,2	0,3	0,5	0,5	0,5	0,25	0,999964	0,5
Density	kg/m³	0,435333	0,639672	0,612617	2,57675	0,915704	0,486272	0,486272	0,31805	0,348576	0,225635	0,612617
Spec Heat @ T	J/(kg*K)	1003,47	1004,48	1004,48	1035,57	1262,85	1235,17	1235,17	1211,94	1211,94	1211,94	1004,48
Spec Heat @ Ts	J/(kg*K)	1003,19	1004,27	1004,15	1034,71	1260,46	1227,82	1227,82	1203,79	1210	1180,2	1004,15
Enthalpy @ T	J/K	-53881,5	-11562,3	-11562,3	234156	1,32642E6	1,08238E6	1,08238E6	917418	917418	917418	-11562,3
Enthalpy @ Ts	J/K	-65581,9	-20492,3	-25276,6	229990	1,30246E6	1,02572E6	1,02572E6	866589	904362	733963	-25276,6
Entropy Function @ T		-0,694205	-0,137787	-0,137787	2,02434	6,16268	5,53318	5,53318	5,04877	5,04877	5,04877	-0,137787
Entropy Function @ Ts		-0,865075	-0,248053	-0,30859	1,99678	6,10475	5,37299	5,37299	4,88762	5,00796	4,43536	-0,30859
Gas Constant	J/(kg*K)	287,05	287,05	287,05	287,05	287,046	287,046	287,046	287,046	287,046	287,046	287,05
Fuel-Air-Ratio		0	0	0	0	0,026148	0,026148	0,026148	0,026148	0,026148	0,026148	0
Water-Air-Ratio		0	0	0	0	0	0	0	0	0	0	0

Fig. 4.8 Tabla con los valores de los estados de un turbofan, modelizado por el programa Gas Turb 11.

```
      FAN : f
Relacion By-pass:                                  3.000000        Kg/s
Cantidad de flujo total a la entrada:              42.524100       Kg/s
Area de la seccion total de entrada:               0.638541        m^2

Temperatura a la entrada:                          244.442000      k
Presion a la entrada:                              34509.000000    Pa
Mach a la entrada:                                 0.500000
Cantidad de flujo reducido de aire a la entrada:   0.030199

Trabajo del fan:                                   84660.869502    J/kg
Relacion de compresion:                            1.650000
Eficiencia politropica:                            0.900000
Eficiencia isentropica:                            0.892696

      FLUJO INTERNO :
Cantidad de flujo a la entrada:                    10.631025       Kg/s
Area de la seccion de entrada:                     0.159636        m^2

Temperatura a la salida:                           286.561836      k
Presion a la salida:                               56939.850000    Pa
Mach a la salida:                                  0.400000
Cantidad de flujo reducido de aire a la salida:    0.025423
Canditadad de flujo a la salida:                   10.631025       Kg/s
Area de la seccion de salida:                      0.124322        m^2

Trabajo interior:                                  42330.434751    J/kg

      FLUJO EXTERNO :
Cantidad de flujo a la entrada:                    31.893075       Kg/s
Area de la seccion de entrada:                     0.478906        m^2

Temperatura a la salida:                           286.561836      k
Presion a la salida:                               56939.850000    Pa
Mach a la salida:                                  0.500000
Cantidad de flujo reducido de aire a la salida:    0.030172
Canditadad de flujo a la salida:                   31.893075       Kg/s
Area de la seccion de salida:                      0.314269        m^2

Trabajo exterior:                                  42330.434751    J/kg
```

Fig. 4.9 Resultados para un ejemplo de fan 2.

Como se puede ver en la figura 4.9 los valores en este caso se aproximan más entre ambos programas que en el ejemplo del compresor, por lo que se refuerza la teoría de alguna posible interferencia de algún parámetro en el caso anterior.

Por otro lado se pueden comporbar los conceptos teóricos mediante las siguientes gráficas, en las cuales se puede apreciar como varian las áreas de las secciones de entrada y los flujos másicos en función de la relación de by-pass. Por tanto se puede ver que para ésta modelización de fan simplemente lo tratamos como un divisor de flujos másicos en el cual según el aumento de la relación de by-pass el flujo de corriente caliente el flujo de dicha corriente diminuye mientras que el frio crece, tal y como se ve en la figura 4.10. En ésta misma figura comprobamos como la suma de flujos de las dos corrientes suma el total para todo el fan, en este caso 100 kg/s.

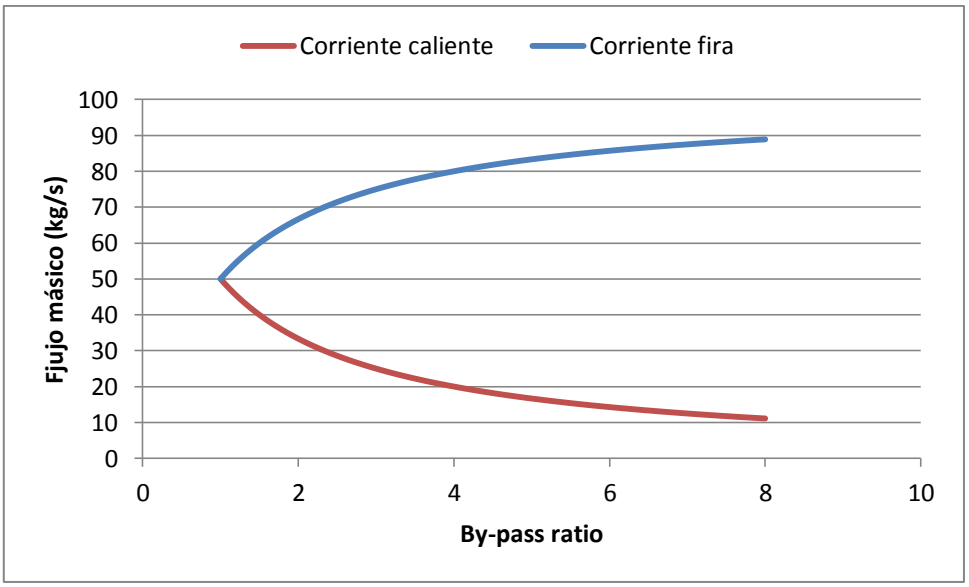


Fig. 4.9 Gráfica de relación de flujos másicos en función del la relación de by-pass en un fan.

Concretando más los resultados de la gráfica de la figura 4.10, se observa como con las relaciones de áreas a la entrada de del fan para cada corriente se obtiene una forma igual al anterior, i por lo tanto, el área de entrada que atraviesa la corriente caliente también disminye mientras que la fría aumenta. Del mismo modo el área total comprovaos que en todo momento es la suma de ambas, que en este caso vale 0.58 m^2.

Finalmente, comentar que si se llevára a cabo otro tipo de modelización para este componente en la cual la relación de compresión no fuera igual para las dos reginoes del compresor, se obtendrían comportamientos de rendimientos y trabajo distientos. Por tanto, también se tendiran variaciónes de temperaturas y presiones a la salida de ésta nueva modelización.

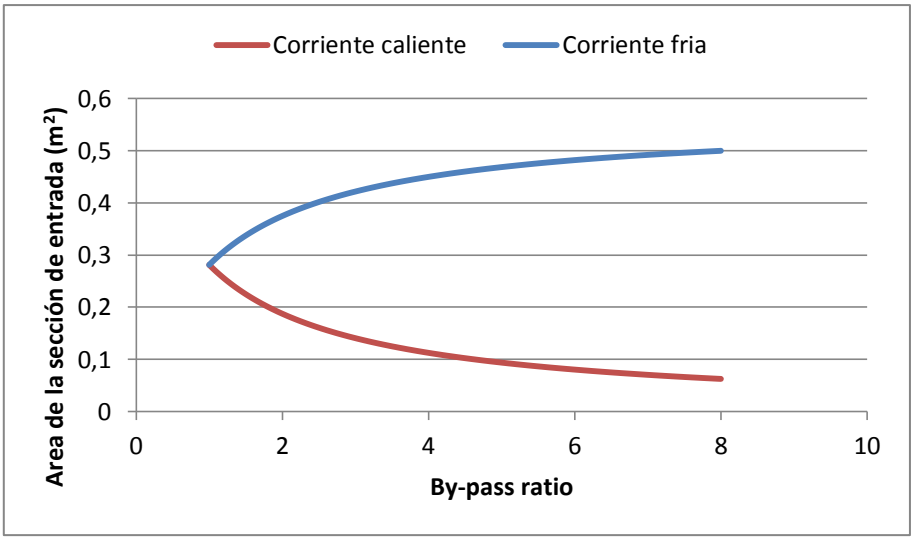


Fig. 4.10 Gráfica de relación de áreas de entrada en función del la relación de by-pass.

CAPÍTULO 5. MOTORES COMPLETOS

Una vez realizados los módulos que permiten diseñar individualmente los componentes hay que comprobar que éstos son capaces de unirse con módulos externos realizados paralelamente, pero que todos juntos forman parte de un proyecto común más amplio y ambicioso. Para ello será necesario unirlos con criterio y es por eso que se implementaran diversos tipos de motor modelo, puesto que hay que recordar que el objetivo del programa era ser capaz de montar los diferentes módulos en el orden que el usuario requiera.

En este trabajo se ha montado un turbojet ideal el cual estará integrado por un difusor, un compresor, una cámara de combustión, una turbina y una tobera. Matizar que dicha tobera será concretamente del tipo convergente-divergente y se encargara de expandir totalmente el gas antes de expulsarlo a la misma presión atmosférica, como se aprecia en la siguiente imágen. El hecho de que la presión de salida de la tobera sea la atmosférica es debido a que se supone como una tobera adaptada.

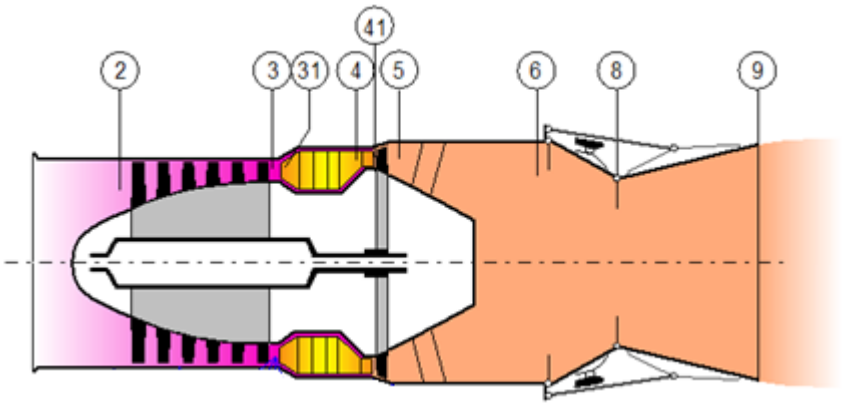


Fig. 5.1 Dibujo esquemático de un turbojet con tobera convergente-divergente.

Este primer motor será montado con componentes de nivel de modelización 0 por tanto se tratara de un motor ideal el cual seguirá completamente el ciclo de Brayton, explicado en el segundo capitulo.

Los problemas más frecuentes que nos encontraremos al ensamblar los motores serán problemas de correspondencia de unidades, por ello es esencial ponerse de acuerdo y usar de manera normalizada el sistema internacional de unidades. Otro tipo de problema es lograr conexionar bien todos los datos con las variables de la funciones, así como establecer de manera clara las relaciones entre las entradas y las salidas para los diferentes componentes.

Ensamblar estos módulos no solo requiere de ellos mismos, si no de módulos que ayuden a darnos la información necesaria para su correcto funcionamiento, como puede ser un módulo atmósfera que le indique al motor según la altura en que se encuentra que temperatura y presión total tiene a la entrada. Otro

módulo añadido permitirá tener resultados genéricos, como pudiera ser el empuje total.

5.1. Resultados para un turbojet ideal

Una vez todo esta montado, se puede comprobar los valores del motor con el *Gas Turb 11* siempre y cuando se establezcan todas las eficiencias con el valor de uno, es decir, que el motor no tenga pérdidas de ningún tipo. Así pues, en la siguiente figura se puede observar dichos valores.

```
          W          T          P        WRstd
Station  kg/s        K         kPa        kg/s        FN        =    182,80 kN
 amb                268,65     70,109                 TSFC      =   27,0771 g/(kN*s)
  1     247,180     288,03     89,429                 FN/W2     =    739,54 m/s
  2     247,180     288,03     89,429    280,000
  3     247,180     580,20   1073,148     33,117      Prop Eff  =    0,3535
 31     247,180     580,20   1073,148                 eta core  =    0,5124
  4     252,130    1300,00   1073,148     50,564
 41     252,130    1300,00   1073,148     50,564      WF        =   4,94968 kg/s
 49     252,130    1058,12    451,087                 s NOx     =   0,21140
  5     252,130    1058,12    451,087    108,527      XM8       =    1,0000
  6     252,130    1058,12    451,087                 A8        =    0,4777 m²
  8     252,130    1058,12    451,087    108,527      P8/Pamb   =    6,4341
Bleed     0,000     580,20   1073,145                 WBld/W2   =   0,00000
-------------------------------------------          Ang8      =     20,00 °
P2/P1 = 1,0000  P4/P3 = 1,0000  P6/P5 1,0000          CD8       =    0,9600
Efficiencies:   isentr  polytr    RNI    P/P         W_NGV/W2  =   0,00000
 Compressor     1,0000  1,0000  0,883 12,000         WCL/W2    =   0,00000
 Burner         1,0000                1,000         Loading   =    100,00 %
 Turbine        1,0000  1,0000  1,816  2,379        e45 th    =   1,00000
-------------------------------------------          far7      =   0,02002
Spool mech Eff  1,0000  Nom Spd    14000 rpm         PWX       =      0,00 kW
-------------------------------------------

hum [%]     war0        FHV      Fuel
   0,0   0,00000     43,100     Generic
```

Fig. 5.2 Resultados del Gas Turb 11 para un turbojet completo

Para poder comparar con nuestro programa, se ha tenido que buscar un cociente entre el flujo de combustible y el flujo de aire de 0.02 para poder satisfacer la relación estequiométrica a la cual esta sometida nuestro modelo de turborreactor ideal.

Una vez realizado esto, se podrá comprobar que los valores para de presiones y temperaturas coincidirán muy rigurosamente hasta la entrada a la cámara de combustión, donde debido a que el *Gas Turb 11* no sigue la relación estequimétrica, se produce una diferencia en la temperatura. Consecuentemente los componentes siguientes arrastrarán dicho error. No obstante si se pasa por alto dicho error y se comparan los términos generales del motor habiendo igualado la relación estequiométrica al máximo, se puede ver en la figura 5.3 que los resultados de empuje y TSFC (denominado en nuestro programa consumo especifico de fuel por unidad de empuje) se ajustan bastante.

```
      PARAMETROS DEL MOTOR :
Empuje:                                  185024.689891    N
Potencia de propulsion:                  37004937.978211  W
Eficiencia de propulsion:                0.336888
Eficiencia de conversion de energia:     0.336888
Eficiencia global:                       0.182677
Consumo especifico de fuel de empuje:    25.402015        g/kNs
Consumo especifico de fuel de propulsion: 0.000000        Kg/Ws
```

Fig. 5.3 Resultados de ejemplo para un turbojet completo.

Otro tipo de prueba para testear la modelización realizada del truborreactor puede consistir en la creación de la figura 5.4. En éste gráfico realizado, se ve como evoluciona la temperatura de entrada a la turbina, es decir, la de salida de la cámara de combustión con respecto de la relación de compresión del compresor, el empuje específico y el consumo específico de combustible para una altura de 9000m, con una velocidad de 243 m/s y un flujo másico de entrada al combustor de 95 kg/s. Con ello podemos constatar que para un empuje mayor es necesario un consumo más grande y que la temperatura crecerá linialmente. Peró como muestra el grafico si se aumenta la relación de compresión se conseguirá un menor consumo para una misma temperatura.

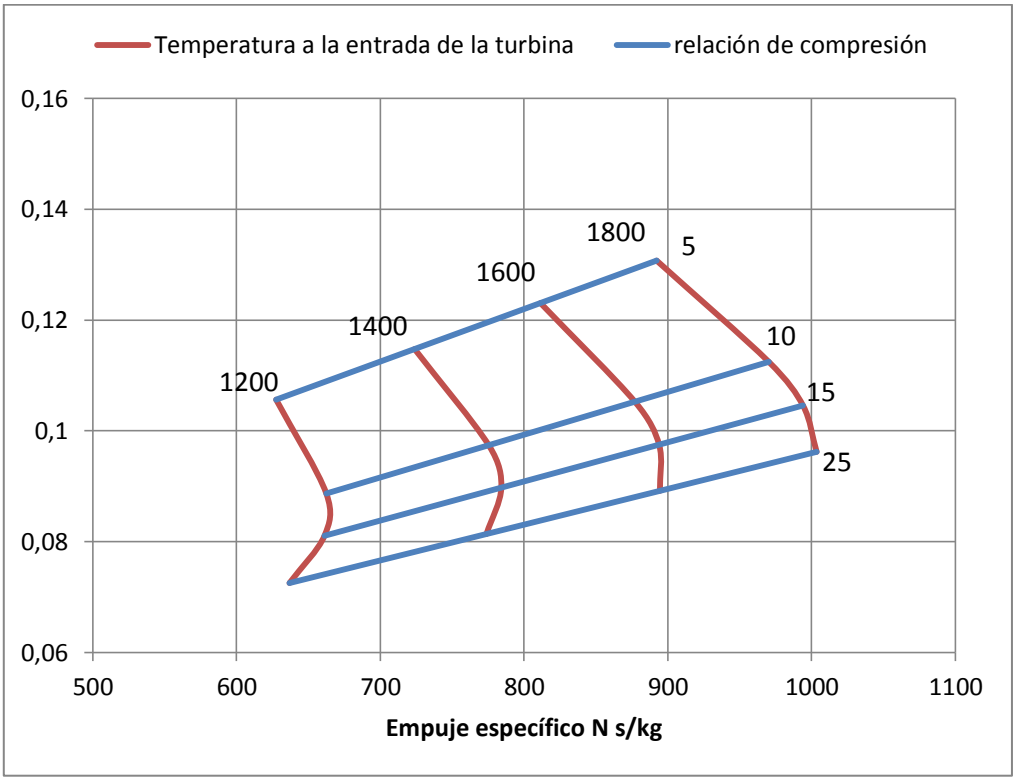


Fig. 5.3 Resultados de ejemplo para un turbojet completo.

CAPÍTULO 6. FUTURAS AMPLIACIONES

En el siguiente capítulo se hablará de los posibles caminos a considerar para mejorar y ampliar el programa. Para ello se ha dividido el capitulo en dos partes, el desarrollo teórico de nuevos conceptos en los componentes formantes de la turbomaquinaria, y por otro lado, nuevos posibles ensamblajes a realizar.

6.2. Ampliaciones en la turbomaquinária

En cuanto a los componentes turbomecánicos, futuras ampliaciones de grado inmediato podrían ser la programación de las magnitudes estáticas, o hacer un estudio riguroso para hacer los módulos y los bancos de prueba más robustos, por ejemplo, mejorando los valores iniciales en los cuales se empieza a iterar o aplicando la optimización del programa a través de las librerías BLAS y LAPAK.

Si bien es cierto que se podría discutir estos temas, para este trabajo se ha preferido seguir avanzando e introducir nuevos conceptos teóricos para el siguiente grado de modelización.

Como se ha visto en este trabajo se han ido tratando las eficiencias politrópicas e isentrópicas como datos que aparecen definidos por el usuario, pero lo cierto es que esto no es así. Estos rendimientos se pueden modelizar a través del rendimiento isentrópico en función del flujo másico reducido, la relación de compresión y la velocidad de giro del compresor y turbina, N. La complicación de dicha dependencia recae en que no existe una fórmula teórica que las relacione directamente, es por ello que se usaran los mapas compresores, los cuales contienen los datos empíricos según el tipo de motor. Es decir, se adquieren los datos reales en un banco de prueba real en el cual se ensaya diversas veces el motor modelo hasta obtener datos suficientes como para generar dichas gráficas.

Estos mapas compresor se representaran como en la figura 6.1, donde se puede observar como el rendimiento isentrópico varia en función de la curva de velocidad de rotación. Apuntar que a pesar que la relación de compresión este definida en un eje también se podrá definir como función del flujo reducido y de la curva de velocidad de rotación.

Para las turbinas el mapa compresor tendrá una representación distinta, como se puede ver en la figura 6.2. Al no depender el flujo másico adimensional de la velocidad adimensional se ve como esto implica una dependencia directa de la relación de compresión con el flujo másico quedando representado con una sola curva.

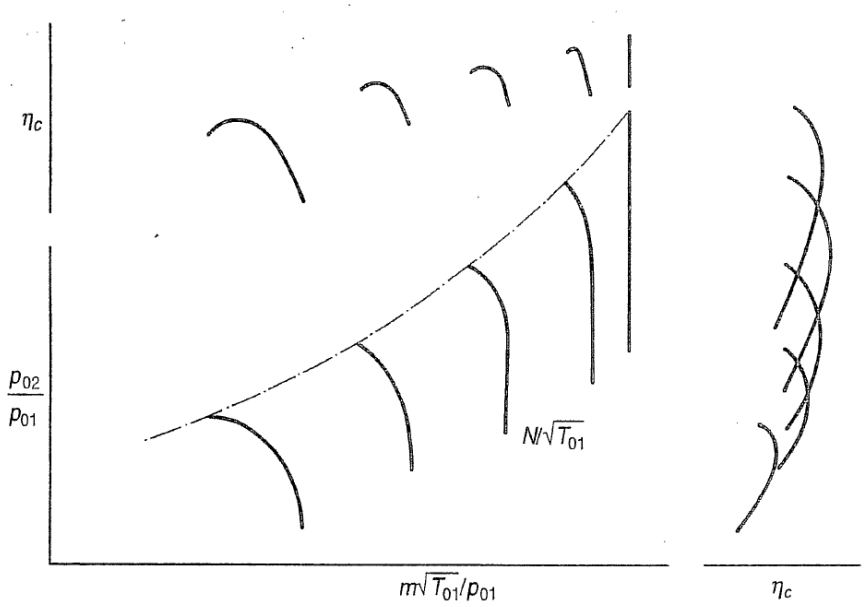


Fig. 6.1 Modelo de mapa compresor para un compresor, extraído de la referencia [1]

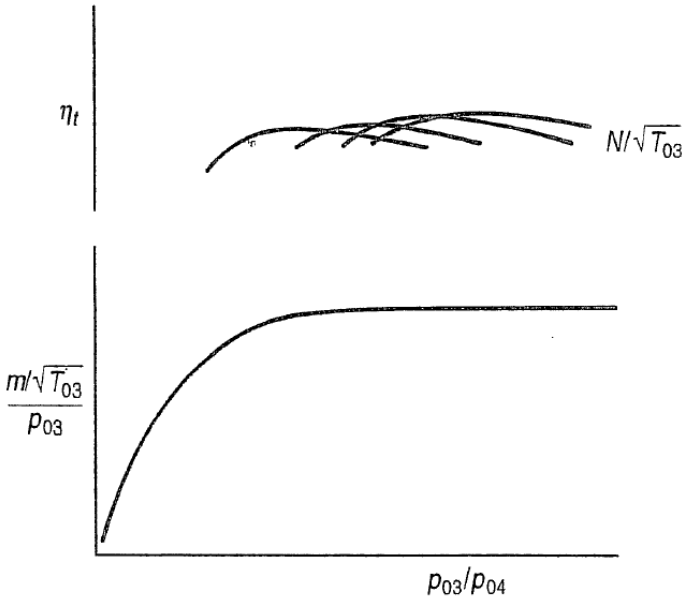


Fig. 6.2 Modelo de un mapa compresor para una turbina, extraído de la referencia [1]

A pesar de que estas gráficas permiten ver claramente todas las curvas representadas, el formato más típico y usado de manera más extendida se puede ver en la figura 6.3 donde se integran en un solo par de ejes diversos parámetros característicos de los componentes.

La manera para programar todo esto será a partir de unas tablas donde se hallaran diversos puntos de las gráficas, un ejemplo de puntos de estas tablas se puede ver a continuación:

Tabla 6.1 Datos de un mapa compresor.

$Pout/Pin$	$\dot{m}\sqrt{T_{in}}/P_{in}$	η_c
5.0	329	0.84
4.5	339	0.79
1.0	342	0.75

Por tanto, dichas tablas se guardaran en un fichero, de texto u otra extensión de la cual se pueda extraer la información, y el programa deberá recorrerlo en función de los parámetros que se tengan y extraer los valores que dependan de éstos. En el caso, que por ejemplo dicho punto no éste definido directamente en la tabla, el programa tendrá que implementar una función que se encargue de interpolar entre valores conocidos. Por tanto, un estudio a tener en cuenta en futuras modelizaciones será que tipo de interpolación se tendría que aplicar, puesto que existen diversas.

Las interpolaciones más conocidas son la lineal y la cuadrática. En la interpolación lineal se usaran 2 puntos por tanto se asumirá que la curva es una línea recta, esto comporta un resultado poco menos realista en el caso de puntos a distancias muy grande entre si, pero por el contrario implica más facilidad en el momento de ser calculado. Por otra parte la interpolación cuadrática dará resultados más fiables pero requerirá un mayor esfuerzo de cálculo.

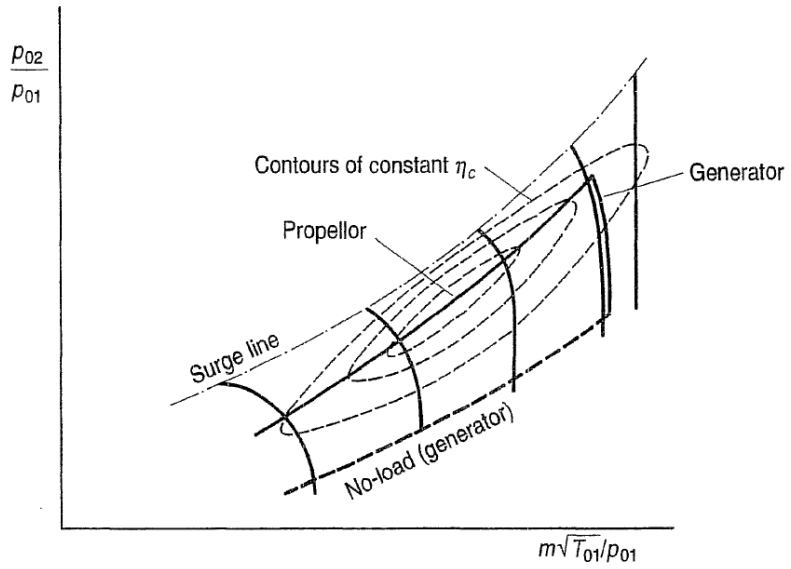


Fig. 6.3 Manera más genérica de representar el mapa compresor. Gráfico extraído de la referencia [1]

CONCLUSIONES

En este trabajo se ha realizado una gran cantidad de materia y se ha podido comprobar que es un proyecto muy amplio, lleno de posibilidades y del cual se puede extraer buenos resultados.

Si se hace un balance general del trabajo realizado se verá como se han conseguido los objetivos propuestos para la materia tratada. Se ha podido primeramente, asentar conceptos termodinámicos básico e importantes, así como familiarizarse bien con los motores y la turbomaquinaria. Igualmente, este ha servido para repasar la programación de objetos y se ha podido crear un primer programa, que ha sido un buen ejercicio para coger práctica a la hora de programar así como iniciarse en los conceptos del método numérico, y comprobar la efectividad del método de Newton-Raphson.

A continuación, de cada módulo realizado con éxito se ha podido realizar una introducción conceptual, una modelización en C++ y un análisis de los resultados. En dicho análisis de los resultados se ha mirado por el correcto funcionamiento en diversos aspectos, como que se cumpla lo esperado en los conceptos teóricos, por otra parte la fiabilidad de los resultados comparándolos con ejemplos ya resueltos o bien el Gas Turb 11, y de este modo se ha llegado a obtener una calidad muy buena en la resolución de los valores finales. Igualmente, también se ha podido comprobar que dicha resolución se puede modificar según le interese al usuario graduando los valores iniciales para el método numérico o graduando el numero de decimales con el que se evalúan las funciones, puesto no es lo mismo evaluar a 10^{-2} que a 10^{-6}.

Además de obtener unos buenos resultados de manera individual, también se ha visto que los resultados también son buenos en el ensamblaje en motores de los diversos módulos, y como es posible realmente ensamblar dichos módulos. Por tanto, es en éste capítulo donde se puede apreciar definitivamente que realizar dicha herramienta de diseño y simulación es posible. Puesto que del mismo modo que se realiza un motor a reacción ideal, se podrían obtener motores como el turbofán, el turbohélice o bien, el diseño que deseara el usuario.

Si bien es cierto, que se trata de un proyecto ambicioso, el poder llegar algún día a introducir modelizaciones que dejen de tratar los componentes como cajas negras, para pasar a tener piezas mecánicas en el interior de estas, también supone una motivación adicional si se aprecia la materia de motores. Al definirse los módulos de esta nueva manera comportará un cambio conceptual importante, debido al modo en el que interaccione el flujo de aire con las piezas internas. Por tanto, en esos grados de modelización se podrá diferenciar entre componentes axiales y centrífugos. Además para poderse analizar correctamente habrá que hacer un uso mayor de las teorías aerodinámicas. Para poder continuar con todas estas ampliaciones y seguir creando motores, gracias a la creación de este trabajo se tendrá una buena base solida de donde partir.

Por otro lado, también se podría empezar a plantear un caminó de evolución paralelo a la programación más teórica, a nivel de funcionamiento, en la cual se buscaría la optimización del programa, y sobretodo para los motores ensamblados, la creación de una buena interfaz gráfica con la que el usuario se pueda encontrar cómodo.

A nivel personal, reconocer que ha sido un trabajo motivador e instructivo, en el cual se ha podido repasar y agrupar muchos conceptos de la carrera, y en el cuál se podría seguir trabajando si se dispusiera de más tiempo.

BIBLIOGRAFIA

[1] Cohen, H., Rogers, G. F. C., Saravanamutto, H. I. H., *Teoria de las turbinas de gas*, Marcombo Boixareu Editores, Barcelona, 1983.

[2] Dixon, S. L., “Introduction: Dimensional Analysis: Similitude”, Cap. 1 en *Fluid Mechanics, Thermodynamics of Turbomachinery*, Elsevier Butterworth-Heinemann editorial, pág. 1-23, Oxford, 1998.

[3] Baskharone, E.A., *Principles of Turbomachinery in Air Breathing Engines,* Cambrige University Pres, Cambrige, 2006.

[4] Hoffman, J. D., *Numerical Methods for Engineers and Scientists*, Marcel Dekker. Inc., Basel, 2001.

[5] Cengel, Y.A., Boles, M. A., *Termodinámica,* McGraw-Hill, Méjico, 2009.

[6] Arrègle, J., Broatch, J. A., Galindo, J., Luján, J. m., Pastor, J. M., Payri R., Serrano, J.R., Torregosa, A.J., *Procesos y Tecnología de Máquinas y Motores Térmicos,* Universidad Politécnica de Valencia, Valencia.

[7] Roll-Royce team, *The Jet Engine*, Rolls-Royce., Derby, 1986.

[8] http://es.scribd.com/doc/15638680/Metodo-de-NewtonRaphson

[9] http://www.grc.nasa.gov/WWW/k-12/VirtualAero/BottleRocket/airplane/mflchk.html

ANEXOS A: Fichas resumen de los componentes

Modelo de ficha:

Component: ***Modeling degree:***

Images and graphics

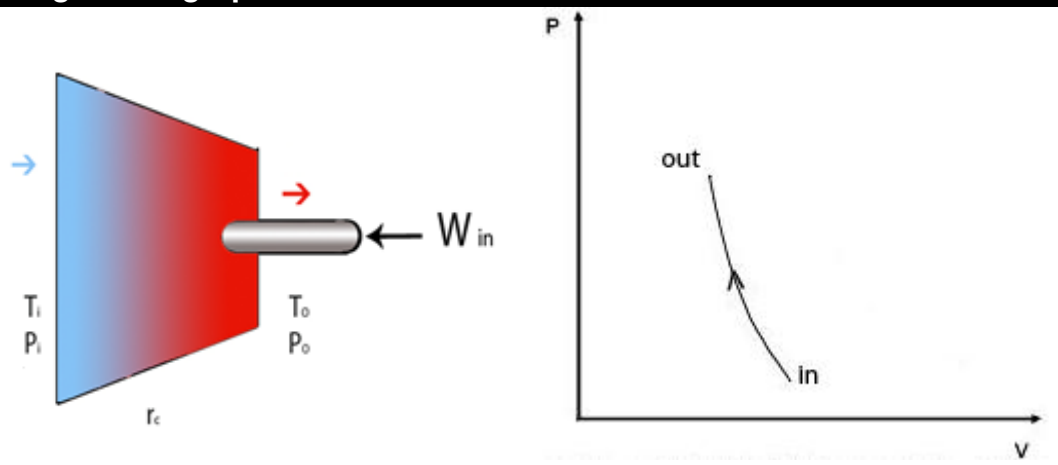
Description and assumptions

Variables	**Formulas**
	Main
	Secondary

Observations for programing

Component: Compressor ***Modeling degree: 0***

Images and graphics



Description and assumptions

This device increases the pressure of a gas by reducing its volume, and rise the energy of the fluid. For that, it takes work from de turbine.

- It is isentropic: Adiabatic and irreversible. Cycle of Brayton.
- Without losses of charge.
- The fluid is a perfect gas with constant specific heats and invariable composition. This fluid is considered stationary.

Variables

- Temperature, T.
- Pressure, P.
- Volume, V.
- Enthalpy, h.
- Isentropic work, W_{in}.
- Pressure ratio, r_c.
- Specific heats, C_p y C_v.
- Adiabatic coefficient, ɣ.
- Gas constant, R.

Formulas

Main

$$P_i^{1-\gamma} T_i^{\gamma} = P_o^{1-\gamma} T_o^{\gamma}$$

$$r_c = \frac{P_o}{P_i}$$

$$W = (h_i - h_o) = C_p(T_o - T_i)$$

Secondary

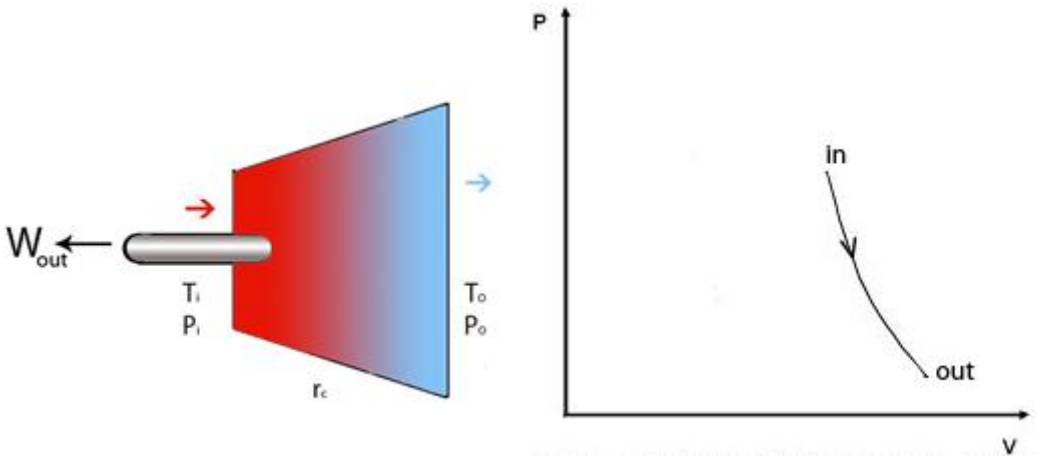
$$\gamma = \frac{C_p}{C_v}$$

$$C_p = C_v + R$$

Observations for programing

Component: Turbine ***Modeling degree: 0***

Images and graphics



Description and assumptions

This device extracts energy from the fluid converting it in work for the shaft. This means an expansion of the gas, so the pressure decrease and the temperature increase.

- It is isentropic: Adiabatic and irreversible. Cycle of Brayton.
- Without losses of charge.
- The fluid is a perfect gas with constant specific heats and invariable composition. This fluid is considered stationary.

Variables

- Temperature, T.
- Pressure, P.
- Volume, V.
- Enthalpy, h.
- Isentropic work, W_{out}.
- Pressure ratio, r_c.
- Specific heats, C_p y C_v.
- Adiabatic coefficient, ɣ.
- Gas constant, R.

Formulas

Main

$$P_i^{1-\gamma} T_i^{\gamma} = P_o^{1-\gamma} T_o^{\gamma}$$

$$r_c = \frac{P_o}{P_i}$$

$$W = (h_i - h_o) = C_p(T_o - T_i)$$

Secondary

$$\gamma = \frac{C_p}{C_v}$$

$$C_p = C_v + R$$

Observations for programing

Component: Fan ***Modeling Degree: 0***

Images and graphics

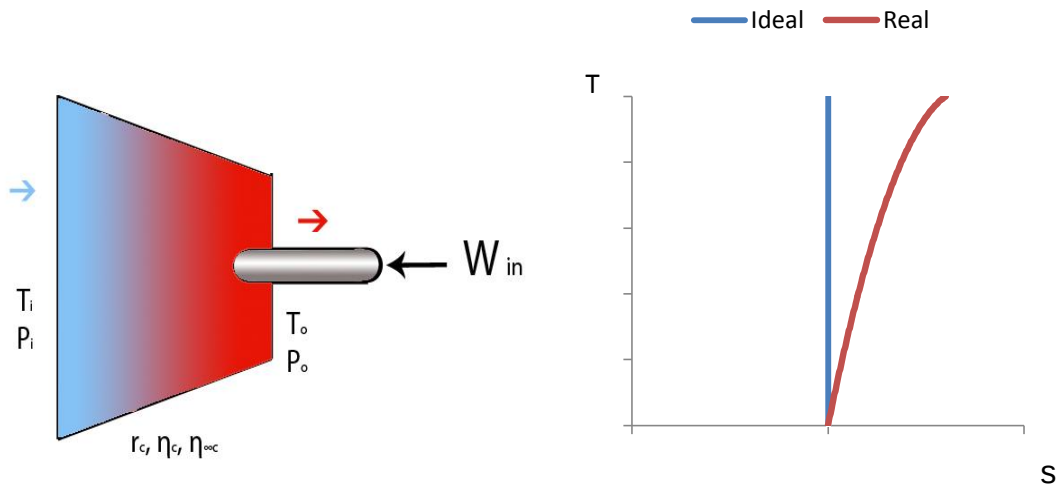
Description and assumptions
In this case, the model of a fan it's equal than a compressor model in 0 degree.

Variables	**Formulas**
	Main
	Secondary

Observations for programing

Component: Compressor ***Modeling degree: 1***

Images and graphics



Description and assumptions

This device increases the pressure of a gas by reducing its volume, and raises the energy of the fluid. For that, it takes work from de turbine.

- Reversible.
- Without losses of charge, but with polytropic and isentropic efficiencies.
- The fluid is a perfect gas with constant specific heats and invariable composition. This fluid is considered stationary.

Variables

- Temperature, T.
- Pressure, P.
- Polytropic efficiency, $\eta_{\infty c}$
- Isentropic efficiency, $\eta_{\infty c}$
- Volume, V.
- Entropy, S.
- Enthalpy, h.
- Isentropic work, W_{in}.
- Pressure ratio, r_c.
- Specific heats, C_p y C_v.
- Adiabatic coefficient, ɣ.
- Gas constant, R.

Formulas

Main

$$\frac{T_{o\ ideal}}{T_{in}} = \left(\frac{P_o}{P_{in}}\right)^{(\gamma-1)/\gamma}$$

$$r_c = \frac{P_o}{P_i}$$

$$Wreal = (h_i - h_o) = C_p(T_{o\ real} - T_i)$$

$$Wideal = (h_i - h_o) = C_p(T_{o\ ideal} - T_i)$$

$$\eta_c = \frac{Wideal}{Wreal}$$

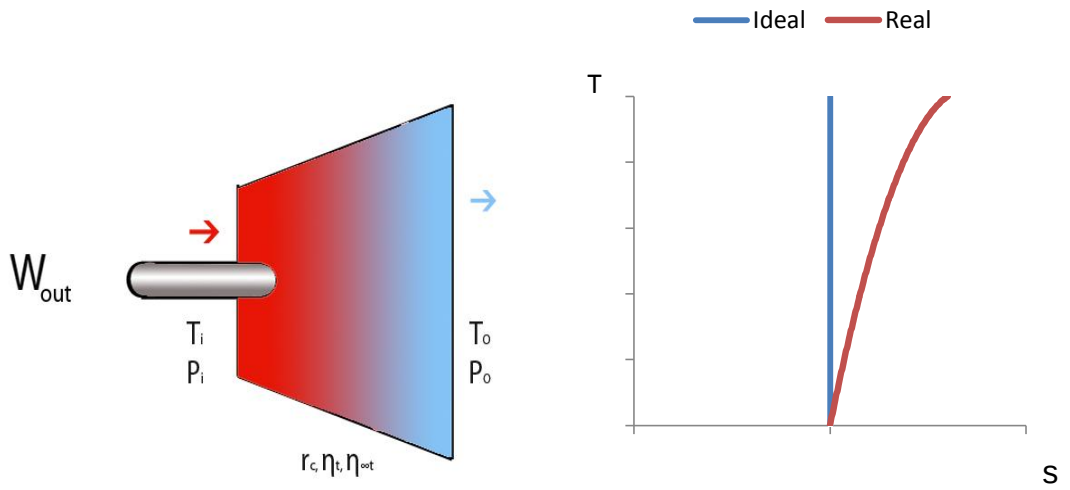
$$\frac{T_{o\ real}}{T_i} = \left(\frac{P_o}{P_i}\right)^{(\gamma-1)/\gamma\eta_{\infty c}}$$

	Secondary
	$\gamma = \frac{C_p}{C_v}$ $C_p = C_v + R$

Observations for programing

Component: Turbine ***Modeling degree: 1***

Images and graphics



Description and assumptions

This device extracts energy from the fluid converting it in work for the shaft. This means an expansion of the gas, so the pressure decrease and the temperature increase.

- Reversible.
- Without losses of charge, but with polytropic and isentropic efficiencies.
- The fluid is a perfect gas with constant specific heats and invariable composition. This fluid is considered stationary.

Variables

- Temperature, T.
- Pressure, P.
- Polytropic efficiency, $\eta_{\infty c}$
- Isentropic efficiency, $\eta_{\infty c}$
- Volume, V.
- Entropy, S.
- Enthalpy, h.
- Isentropic work, W_{in}.
- Pressure ratio, r_c.
- Specific heats, C_p y C_v.
- Adiabatic coefficient, ɣ.
- Gas constant, R.

Formulas

Main

$$\frac{T_{o\ ideal}}{T_i} = \left(\frac{P_o}{P_i}\right)^{(\gamma-1)/\gamma}$$

$$r_c = \frac{P_o}{P_i}$$

$$Wreal = (h_i - h_o) = C_p(T_{o\ real} - T_i)$$

$$Wideal = (h_i - h_o) = C_p(T_{o\ ideal} - T_i)$$

$$\eta_c = \frac{Wreal}{Wideal}$$

$$\frac{T_o}{T_i} = \left(\frac{P_o}{P_i}\right)^{\eta_{\infty c}(\gamma-1)/\gamma}$$

	Secondary
	$\gamma = \frac{C_p}{C_v}$ $C_p = C_v + R$

Observations for programing

Component: Fan ***Modeling Degree: 1***

Images and graphics

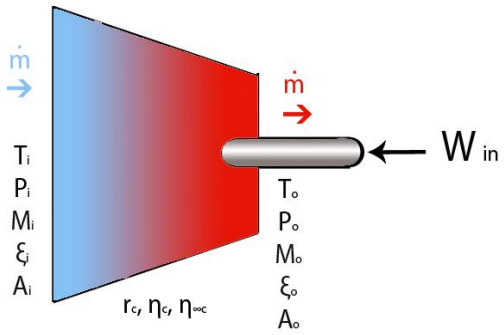
Description and assumptions
In this case, the model of a fan it's equal than a compressor model in 0 degree.

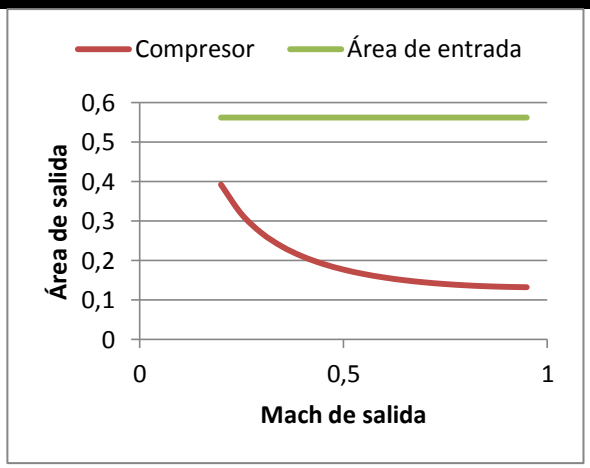
Variables	**Formulas**
	Main
	Secondary

Observations for programing

Component: Compressor ***Modeling degree: 2***

Images and graphics





Description and assumptions

This device increases the pressure of a gas by reducing its volume, and rise the energy of the fluid. For that, it takes work from de turbine.

- Reversible.
- Without losses of charge, but with polytropic and isentropic efficiencies.
- The mass flow is considered constant.
- The fluid is a perfect gas with constant specific heats and invariable composition. This fluid is considered stationary.

Variables

- Total temperature, T, T_{real}.
- Static temperature, T_{ideal}
- Pressure, P.
- Static pressure, P_{est}
- Polytropic efficiency, $\eta_{\infty c}$
- Isentropic efficiency, $\eta_{\infty c}$
- Mach, M.
- Velocity, v
- Mass flux, $\dot{m}$.
- Corrected mass flux, ξ.
- Volume, V.
- Entropy, S.
- Enthalpy, h.
- Isentropic work, W_{in}.
- Pressure ratio, r_c.

Formulas

Main

$$\frac{T_{o\ ideal}}{T_{in}} = \left(\frac{P_o}{P_{in}}\right)^{(\gamma-1)/\gamma}$$

$$r_c = \frac{P_o}{P_i}$$

$$Wreal = (h_i - h_o) = C_p(T_{o\ real} - T_i)$$

$$Wideal = (h_i - h_o) = C_p(T_{o\ ideal} - T_i)$$

$$\eta_c = \frac{Wideal}{Wreal}$$

$$\frac{T_{o\ real}}{T_i} = \left(\frac{P_o}{P_i}\right)^{(\gamma-1)/\gamma\eta_{\infty c}}$$

<table>
<tr>
<td rowspan="3">

- Specific heats, C_p y C_v.
- Adiabatic coefficient, ɣ.
- Gas constant, R.

</td>
<td>

$$\xi = M\sqrt{\frac{\gamma}{R}}\left(1+\frac{(\gamma-1)}{2}M^2\right)^{\frac{-(\gamma+1)}{2(\gamma-1)}}$$

$$\xi = \frac{\dot{m}\sqrt{T}}{P\,A}$$

</td>
</tr>
<tr>
<td>Secondary</td>
</tr>
<tr>
<td>

$$\gamma = \frac{C_p}{C_v}$$

$$C_p = C_v + R$$

$$\dot{m}_{in} = \dot{m}_{out}$$

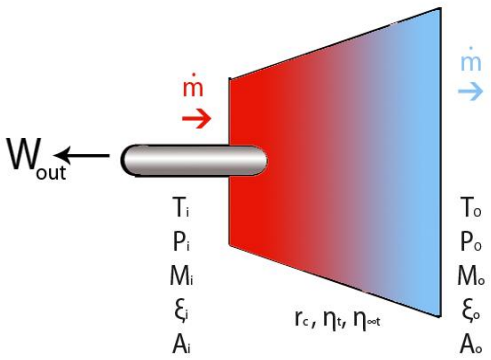
$$M = v/\sqrt{\gamma RT}$$

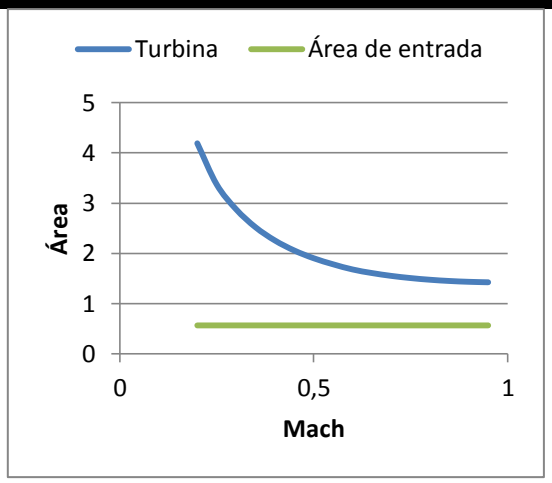
</td>
</tr>
</table>

Observations for programing

Component: Turbine ***Modeling degree: 2***

Images and graphics





Description and assumptions

This device extracts energy from the fluid converting it in work for the shaft. This means an expansion of the gas, so the pressure decrease and the temperature increase.

- Reversible.
- Without losses of charge, but with polytropic and isentropic efficiencies.
- The mass flow is considered constant.
- The fluid is a perfect gas with constant specific heats and invariable composition. This fluid is considered stationary.

Variables

- Total temperature, T, T_{real}.
- Static temperature, T_{ideal}
- Pressure, P.
- Static pressure, P_{st}
- Polytropic efficiency, $\eta_{\infty c}$
- Isentropic efficiency, $\eta_{\infty c}$
- Mach, M.
- Velocity, v
- Mass flux, $\dot{m}$.
- Corrected mass flux, ξ.
- Volume, V.
- Entropy, S.
- Enthalpy, h.
- Isentropic work, W_{in}.

Formulas

Main

$$\frac{T_{o\ ideal}}{T_i} = \left(\frac{P_o}{P_i}\right)^{(\gamma-1)/\gamma}$$

$$r_c = \frac{P_o}{P_i}$$

$$Wreal = (h_i - h_o) = C_p(T_{o\ real} - T_i)$$

$$Wideal = (h_i - h_o) = C_p(T_{o\ ideal} - T_i)$$

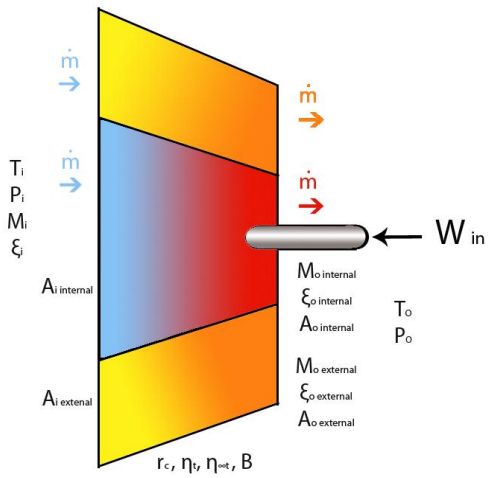
$$\eta_c = \frac{Wreal}{Wideal}$$

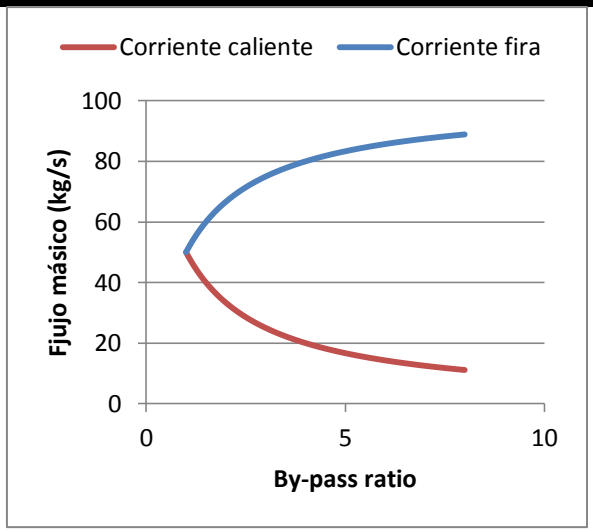
• Pressure ratio, r_c. • Specific heats, C_p y C_v. • Adiabatic coefficient, ɣ. • Gas constant, R.	$$\frac{T_o}{T_i} = \left(\frac{P_o}{P_i}\right)^{\eta_{\infty c}(\gamma-1)/\gamma}$$ $$\xi = M\sqrt{\frac{\gamma}{R}}\,(1+\frac{(\gamma-1)}{2}M^2)^{\frac{-(\gamma+1)}{2(\gamma-1)}}$$ $$\xi = \frac{\dot{m}\sqrt{T}}{P\,A}$$ $$T_0 = T_{st}\,(1+\left(\frac{(\gamma-1)}{2}M^2\right))$$ $$P_0 = P_{st}(1+\left(\frac{(\gamma-1)}{2}M^2\right))^{\gamma/(\gamma-1)}$$
	Secondary
	$$\gamma = \frac{C_p}{C_v}$$ $$C_p = C_v + R$$ $$\dot{m}_{in} = \dot{m}_{out}$$ $$M = v/\sqrt{\gamma RT}$$

Observations for programing

Component: Fan ***Modeling Degree: 2***

Images and graphics





Description and assumptions

This device extracts energy from the fluid converting it in work for the shaft. This means an expansion of the gas, so the pressure decrease and the temperature increase.

- Reversible.
- Without losses of charge, but with polytropic and isentropic efficiencies.
- The mass flow is considered constant for both flows.
- The two sections of the fan are considered with the same pressure ratio and efficiencies.
- The fluid is a perfect gas with constant specific heats and invariable composition. This fluid is considered stationary.

Variables

- Total temperature, T, T_{real}.
- Static temperature, T_{ideal}
- Pressure, P.
- Static pressure, P_{st}
- Polytropic efficiency, $\eta_{\infty c}$
- Isentropic efficiency, $\eta_{\infty c}$
- By-pass ratio, B
- Mach, M.
- Velocity, v
- Mass flux, $\dot{m}$.

Formulas

Main

$$B = \frac{\dot{m}_{\ corriente\ fira}}{\dot{m}_{\ corriente\ caliente}}$$

$$\dot{m} = \dot{m}_{\ corriente\ fira} + \dot{m}_{\ corriente\ caliente}$$

$$W = W_{\ corriente\ fira} + W_{\ corriente\ caliente}$$

• Corrected mass flux, ξ. • Volume, V. • Entropy, S. • Enthalpy, h. • Isentropic work, W_{in}. • Pressure ratio, r_c. • Specific heats, C_p y C_v. • Adiabatic coefficient, ɣ. • Gas constant, R.	Secondary $\frac{T_{o\ ideal}}{T_i} = (\frac{P_o}{P_i})^{(\gamma-1)/\gamma}$ $r_c = \frac{P_o}{P_i}$ $Wreal = (h_i - h_o) = C_p(T_{o\ real} - T_i)$ $Wideal = (h_i - h_o) = C_p(T_{o\ ideal} - T_i)$ $\eta_c = \frac{Wreal}{Wideal}$ $\frac{T_o}{T_i} = \left(\frac{P_o}{P_i}\right)^{\eta_{\infty c}(\gamma-1)/\gamma}$ $\xi = M\sqrt{\frac{\gamma}{R}}\,(1 + \frac{(\gamma-1)}{2}\,M^2)^{\frac{-(\gamma+1)}{2(\gamma-1)}}$ $\xi = \frac{\dot{m}\sqrt{T}}{P\,A}$ $T_0 = T_{st}\,(1 + \left(\frac{(\gamma-1)}{2}\,M^2\right))$ $P_0 = P_{st}(1 + \left(\frac{(\gamma-1)}{2}\,M^2\right))^{\gamma/(\gamma-1)}$ $\gamma = \frac{C_p}{C_v}$ $C_p = C_v + R$ $\dot{m}_{in} = \dot{m}_{out}$ $M = v/\sqrt{\gamma RT}$

Observations for programing
In programing, this modelization the fan is considered like two objects of compressor of the second modeling degree

Anexos B: Código de programación en C++

Método de Newton-Raphson:

Newton.h

```
#define MAX_ROW 20
#define MAX_COLUMN 20

class Newton{
private:


public:

	void Desarrollo (double X [MAX_ROW], double Fx [MAX_ROW], double dFx
[MAX_ROW][MAX_COLUMN], int s, int nFx);

};
```

Newton.cpp

```
#include <stdio.h>
#include <stdlib.h>
#include "f2c.h"
#include "clapack.h"
#include "Newton.h"


void Newton::Desarrollo (double X [MAX_ROW], double Fx [MAX_ROW], double dFx
[MAX_ROW][MAX_COLUMN], int s, int nFx)
{
	int j=0, k=0, t=0;
     /* Ejemplo 3x3 matrix A
     * 76 25 11
     * 27 89 51
     * 18 60 32
     double A[9] = {76, 27, 18, 25, 89, 60, 11, 51, 32};
        */

	double A[MAX_ROW*MAX_COLUMN];
    double b[MAX_ROW];

	while (j < s)
	{
		k=0;
		while (k <= nFx)
		{
		A[t]=dFx[k][j];

		t++;
		k++;
		}
```

```
                b[j]=Fx[j];
                j++;
        }

    integer N = s;//numero columnas A
    integer nrhs = 1;//numero columnas b
    integer lda = nFx+1; // numero filas A
    integer ipiv[MAX_ROW]; // indices del pibote
    integer ldb = nFx+1; // numero filas b
    integer info;

    dgesv_(&N, &nrhs, A, &lda, ipiv, b, &ldb, &info);

       if(info != 0) /* no-succeed */
              fprintf(stderr, "dgesv_ fails %d\n", info);

       integer n=nFx+1, incx=1, incy=1;
       doublereal da=-1;

       daxpy_(&n, &da, b, &incx, X, &incy);     // resta de vectores (-1·x)·y
//     system("pause");
}
```

Compresores:

Modelización 0:

TMc0.h

```
#define MAX_ROW 20
#define MAX_COLUMN 20
#define MAXP 20
typedef char Tpalabra [MAXP];

class TMc0{
private:
       Tpalabra nombre_compresor;
       double Tout,Pout, Tin, Pin, rc, W;
public:
       void pon_nombre (char nombre_compresor[20]);
       Tpalabra* dame_nombre (void);

       void pon_Tout (double ToutC);
       double dame_Tout (void);

       void pon_Pout (double PoutC);
       double dame_Pout (void);

       void pon_Tin (double TinC);
       double dame_Tin (void);

       void pon_Pin (double PinC);
       double dame_Pin (void);

       void pon_rc (double rcC);
       double dame_rc (void);

       void pon_W (double WC);
       double dame_W (void);
```

```
	TMc0(void);

	double Fx1(double ToutC, double PoutC, double TinC, double PinC, double
gamma,TMc0 &TMcompresor0);
	double Fx2(double PoutC, double PinC, double rcC,TMc0 &TMcompresor0);
	double Fx3(double ToutC, double TinC, double WC, double Cp,TMc0
&TMcompresor0);
	int guardar_datos(int err, TMc0 &TMcompresor0);
};
```

TMc0.cpp

```
#include <stdio.h>
#include <string.h>
#include <math.h>
#include "TMc0.h"

void TMc0::pon_nombre(Tpalabra nombre2)
{
	strcpy(nombre_compresor,nombre2);
}
Tpalabra* TMc0:: dame_nombre(void)
{
	return (&nombre_compresor);
}
void TMc0::pon_Tout(double ToutC)
{
	Tout=ToutC;
}
double TMc0::dame_Tout(void)
{
	return (Tout);
}
void TMc0::pon_Pout(double PoutC)
{
	Pout=PoutC;
}
double TMc0::dame_Pout(void)
{
	return (Pout);
}
void TMc0::pon_Tin(double TinC)
{
	Tin=TinC;
}
double TMc0::dame_Tin(void)
{
	return (Tin);
}
void TMc0::pon_Pin(double PinC)
{
	Pin=PinC;
}
double TMc0::dame_Pin(void)
{
	return (Pin);
}
void TMc0::pon_rc(double rcC)
{
```

```
	rc=rcC;
}
double TMc0::dame_rc(void)
{
	return (rc);
}
void TMc0::pon_W(double WC)
{
	W=WC;
}
double TMc0::dame_W(void)
{
	return (W);
}

TMc0::TMc0(void)
{
	pon_nombre("Inicializado");
	pon_Pin(150000); //Pa
	pon_Tin(500); //k
	pon_Pout(400000); //Pa
	pon_Tout(600); //k
	pon_rc(3);
	pon_W(150000);  // J/kg
}
double TMc0::Fx1(double ToutC, double PoutC, double TinC, double PinC, double
gamma,TMc0 &TMcompresor0)
{
	double fx1;

	fx1=(pow (PinC,(1-gamma)) * pow (TinC,gamma))- (pow(PoutC,(1-gamma))* pow
(ToutC,gamma));

	TMcompresor0.pon_Pin(PinC);
	TMcompresor0.pon_Tin(TinC);
	TMcompresor0.pon_Pout(PoutC);
	TMcompresor0.pon_Tout(ToutC);
	return fx1;
}

double TMc0::Fx2(double PoutC, double PinC, double rcC,TMc0 &TMcompresor0)
{
	double fx2;

	fx2= (PoutC-(rcC*PinC));

	TMcompresor0.pon_rc(rcC);
	TMcompresor0.pon_Pin(PinC);
	TMcompresor0.pon_Pout(PoutC);
	return fx2;
}

double TMc0::Fx3(double ToutC, double TinC,double WC, double Cp,TMc0
&TMcompresor0)
{
	double fx3;

	fx3=(WC -(Cp*(ToutC - TinC)));

	TMcompresor0.pon_W(WC);
	TMcompresor0.pon_Tin(TinC);
	TMcompresor0.pon_Tout(ToutC);
```

```
	return fx3;
}

int TMc0::guardar_datos(int err, TMc0 &TMcompresor0)  //guardar datos
{
	FILE*out;
	if (err==1)
		out=fopen("datos.txt", "w");
	else
		out=fopen("datos.txt", "a");

	if(out==NULL)
	{
		return(-1);
	}
	else
	{
		fprintf(out,"%s\n",TMcompresor0.dame_nombre());
		fprintf(out,"%lf\n%lf\n%lf\n%lf\n%lf\n%lf\n",
TMcompresor0.dame_Pin(), TMcompresor0.dame_Tin(), TMcompresor0.dame_Pout(),
TMcompresor0.dame_Tout(), TMcompresor0.dame_rc(),TMcompresor0.dame_W());

		fclose(out);
		return(0);
	}
}
```

Modelización 1:

TMc1.h

```
#define MAX_ROW 20
#define MAX_COLUMN 20
#define MAXP 20
typedef char Tpalabra [MAXP];

class TMc1{
private:
	Tpalabra nombre_compresor;
	double Tout,Pout, Tin, Pin, rc, W, ToutI, WI, ef_poli, ef_isen;
public:
	void pon_nombre (char nombre_compresor[20]);
	Tpalabra* dame_nombre (void);

	void pon_Tout (double ToutC);
	double dame_Tout (void);

	void pon_Pout (double PoutC);
	double dame_Pout (void);

	void pon_Tin (double TinC);
	double dame_Tin (void);

	void pon_Pin (double PinC);
	double dame_Pin (void);

	void pon_rc (double rcC);
	double dame_rc (void);

	void pon_W (double WC);
	double dame_W (void);
```

```
        void pon_ToutI (double ToutIC);
        double dame_ToutI (void);

        void pon_WI (double WIC);
        double dame_WI (void);

        void pon_ef_isen (double ef_isenC);
        double dame_ef_isen (void);

        void pon_ef_poli (double ef_poliC);
        double dame_ef_poli (void);

        TMc1(void);
        double Fx1(double ToutIC, double PoutC, double TinC, double PinC, double
gamma,TMc1 &TMcompresor1);
        double Fx2(double PoutC, double PinC, double rcC,TMc1 &TMcompresor1);
        double Fx3(double ToutC, double TinC, double WC, double Cp,TMc1
&TMcompresor1);
        double Fx4(double ToutIC, double TinC, double WIC, double Cp,TMc1
&TMcompresor1);
        double Fx5(double WC, double ef_isenC, double WIC, TMc1 &TMcompresor1);
        double Fx6(double ToutC, double PoutC, double TinC, double PinC, double
ef_poliC, double gamma,TMc1 &TMcompresor1);
        int guardar_datos(int err, TMc1 &TMcompresor1);
};
```

TMc1.cpp

```
#include <stdio.h>
#include <string.h>
#include <math.h>
#include "TMc1.h"

void TMc1::pon_nombre(Tpalabra nombre2)
{
        strcpy(nombre_compresor,nombre2);
}
Tpalabra* TMc1:: dame_nombre(void)
{
        return (&nombre_compresor);
}
void TMc1::pon_Tout(double ToutC)
{
        Tout=ToutC;
}
double TMc1::dame_Tout(void)
{
        return (Tout);
}
void TMc1::pon_Pout(double PoutC)
{
        Pout=PoutC;
}
double TMc1::dame_Pout(void)
{
        return (Pout);
}
void TMc1::pon_Tin(double TinC)
{
```

```
	Tin=TinC;
}
double TMc1::dame_Tin(void)
{
	return (Tin);
}
void TMc1::pon_Pin(double PinC)
{
	Pin=PinC;
}
double TMc1::dame_Pin(void)
{
	return (Pin);
}
void TMc1::pon_rc(double rcC)
{
	rc=rcC;
}
double TMc1::dame_rc(void)
{
	return (rc);
}
void TMc1::pon_W(double WC)
{
	W=WC;
}
double TMc1::dame_W(void)
{
	return (W);
}

void TMc1::pon_WI(double WIC)
{
	WI=WIC;
}
double TMc1::dame_WI(void)
{
	return (WI);
}
void TMc1::pon_ToutI(double ToutIC)
{
	ToutI=ToutIC;
}
double TMc1::dame_ToutI(void)
{
	return (ToutI);
}
void TMc1::pon_ef_poli(double ef_poliC)
{
	ef_poli=ef_poliC;
}
double TMc1::dame_ef_poli(void)
{
	return (ef_poli);
}
void TMc1::pon_ef_isen(double ef_isenC)
{
	ef_isen=ef_isenC;
}
double TMc1::dame_ef_isen(void)
{
	return (ef_isen);
```

```
}
TMc1::TMc1(void)
{
	pon_nombre("Inicializado");
	pon_Pin(150000); //Pa
	pon_Tin(500); //k
	pon_Pout(400000); //Pa
	pon_Tout(600); //k
	pon_rc(3);
	pon_W(150000);  // J/kg
	pon_WI(160000); // J/kg
	pon_ToutI(600); //k
	pon_ef_poli(0.9);
	pon_ef_isen(0.85);
}
double TMc1::Fx1(double ToutIC, double PoutC, double TinC, double PinC, double
gamma,TMc1 &TMcompresor1)
{
	double fx1;

	fx1=((ToutIC/TinC) - (pow((PoutC/PinC),((gamma-1)/gamma))));

	TMcompresor1.pon_Pin(PinC);
	TMcompresor1.pon_Tin(TinC);
	TMcompresor1.pon_Pout(PoutC);
	TMcompresor1.pon_ToutI(ToutIC);
	return fx1;
}

double TMc1::Fx2(double PoutC, double PinC, double rcC,TMc1 &TMcompresor1)
{
	double fx2;

	fx2= (rcC-(PoutC/PinC));

	TMcompresor1.pon_rc(rcC);
	TMcompresor1.pon_Pin(PinC);
	TMcompresor1.pon_Pout(PoutC);
	return fx2;
}

double TMc1::Fx3(double ToutC, double TinC,double WC, double Cp,TMc1
&TMcompresor1)
{
	double fx3;

	fx3=(1 -((Cp*(ToutC - TinC))/WC));

	TMcompresor1.pon_W(WC);
	TMcompresor1.pon_Tin(TinC);
	TMcompresor1.pon_Tout(ToutC);
	return fx3;
}

double TMc1::Fx4(double ToutIC, double TinC,double WIC, double Cp,TMc1
&TMcompresor1)
{
	double fx4;

	fx4=(1 -((Cp*(ToutIC - TinC))/WIC));

	TMcompresor1.pon_WI(WIC);
```

```
	TMcompresor1.pon_Tin(TinC);
	TMcompresor1.pon_ToutI(ToutIC);
	return fx4;
}

double TMc1::Fx5(double WC, double ef_isenC,double WIC, TMc1 &TMcompresor1)
{
	double fx5;

	fx5=(1 -((ef_isenC*WC)/WIC));

	TMcompresor1.pon_W(WC);
	TMcompresor1.pon_WI(WIC);
	TMcompresor1.pon_ef_isen(ef_isenC);
	return fx5;
}

double TMc1::Fx6(double ToutC, double PoutC, double TinC, double PinC, double
ef_poliC, double gamma,TMc1 &TMcompresor1)
{
	double fx6;

	fx6=((ToutC/TinC) - (pow((PoutC/PinC),((gamma-1)/(ef_poliC*gamma)))));

	TMcompresor1.pon_Pin(PinC);
	TMcompresor1.pon_Tin(TinC);
	TMcompresor1.pon_Pout(PoutC);
	TMcompresor1.pon_Tout(ToutC);
	TMcompresor1.pon_ef_poli(ef_poliC);
	return fx6;
}
int TMc1::guardar_datos(int err, TMc1 &TMcompresor1)  //guardar datos
{
	FILE*out;
	if (err==1)
		out=fopen("datos.txt", "w");
	else
		out=fopen("datos.txt", "a");

	if(out==NULL)
	{
		return(-1);
	}
	else
	{
		fprintf(out,"%s\n",TMcompresor1.dame_nombre());
		fprintf(out,"%lf\n%lf\n%lf\n%lf\n%lf\n%lf\n",
TMcompresor1.dame_Pin(), TMcompresor1.dame_Tin(), TMcompresor1.dame_Pout(),
TMcompresor1.dame_Tout(), TMcompresor1.dame_rc(),TMcompresor1.dame_W());
		fprintf(out,"Ef_poli %lf\nEf_isen %lf\n",
TMcompresor1.dame_ef_poli(), TMcompresor1.dame_ef_isen());
		fclose(out);
		return(0);
	}
}
```

Modelización 2:

TMc2.h

```
#define MAX_ROW 20
#define MAX_COLUMN 20
#define MAXP 20
typedef char Tpalabra [MAXP];

class TMc2{
private:
	Tpalabra nombre_compresor;
	double Tout,Pout, Tin, Pin, rc, W, ToutI, WI, ef_poli, ef_isen, Min, Din,
Mout, Dout, min, mout, Ain, Aout ;
public:
	void pon_nombre (char nombre_compresor[20]);
	Tpalabra* dame_nombre (void);

	void pon_Tout (double ToutC);
	double dame_Tout (void);

	void pon_Pout (double PoutC);
	double dame_Pout (void);

	void pon_Tin (double TinC);
	double dame_Tin (void);

	void pon_Pin (double PinC);
	double dame_Pin (void);

	void pon_rc (double rcC);
	double dame_rc (void);

	void pon_W (double WC);
	double dame_W (void);

	void pon_ToutI (double ToutIC);
	double dame_ToutI (void);

	void pon_WI (double WIC);
	double dame_WI (void);

	void pon_ef_isen (double ef_isenC);
	double dame_ef_isen (void);

	void pon_ef_poli (double ef_poliC);
	double dame_ef_poli (void);

	void pon_Min (double MinC);
	double dame_Min (void);

	void pon_Din (double DinC);
	double dame_Din (void);

	void pon_Mout (double MoutC);
	double dame_Mout (void);

	void pon_Dout (double DoutC);
	double dame_Dout (void);

	void pon_min (double minC);
	double dame_min (void);

	void pon_mout (double moutC);
	double dame_mout (void);
```

```
        void pon_Ain (double AinC);
        double dame_Ain (void);

        void pon_Aout (double AoutC);
        double dame_Aout (void);

        TMc2(void);
        double Fx1(double ToutIC, double PoutC, double TinC, double PinC, double
gamma,TMc2 &TMcompresor2);
        double Fx2(double PoutC, double PinC, double rcC,TMc2 &TMcompresor2);
        double Fx3(double ToutC, double TinC, double WC, double Cp,TMc2
&TMcompresor2);
        double Fx4(double ToutIC, double TinC, double WIC, double Cp,TMc2
&TMcompresor2);
        double Fx5(double WC, double ef_isenC, double WIC, TMc2 &TMcompresor2);
        double Fx6(double ToutC, double PoutC, double TinC, double PinC, double
ef_poliC, double gamma,TMc2 &TMcompresor2);
        double Fx7(double DinC, double MinC, double gamma, double R,TMc2
&TMcompresor2);
        double Fx8(double DoutC, double MoutC, double gamma, double R,TMc2
&TMcompresor2);
        double Fx9(double TinC, double PinC, double DinC, double minC, double
AinC,TMc2 &TMcompresor2);
        double Fx10(double ToutC, double PoutC, double DoutC, double moutC,
double AoutC,TMc2 &TMcompresor2);
        double Fx11(double minC, double moutC, TMc2 &TMcompresor2);
        int guardar_datos(int err, TMc2 &TMcompresor2);
};
```

TMc2.cpp

```
#include <stdio.h>
#include <string.h>
#include <math.h>
#include "TMc2.h"

void TMc2::pon_nombre(Tpalabra nombre2)
{
        strcpy(nombre_compresor,nombre2);
}
Tpalabra* TMc2:: dame_nombre(void)
{
        return (&nombre_compresor);
}
void TMc2::pon_Tout(double ToutC)
{
        Tout=ToutC;
}
double TMc2::dame_Tout(void)
{
        return (Tout);
}
void TMc2::pon_Pout(double PoutC)
{
        Pout=PoutC;
}
double TMc2::dame_Pout(void)
{
        return (Pout);
}
```

```
void TMc2::pon_Tin(double TinC)
{
	Tin=TinC;
}
double TMc2::dame_Tin(void)
{
	return (Tin);
}
void TMc2::pon_Pin(double PinC)
{
	Pin=PinC;
}
double TMc2::dame_Pin(void)
{
	return (Pin);
}
void TMc2::pon_rc(double rcC)
{
	rc=rcC;
}
double TMc2::dame_rc(void)
{
	return (rc);
}
void TMc2::pon_W(double WC)
{
	W=WC;
}
double TMc2::dame_W(void)
{
	return (W);
}
void TMc2::pon_WI(double WIC)
{
	WI=WIC;
}
double TMc2::dame_WI(void)
{
	return (WI);
}
void TMc2::pon_ToutI(double ToutIC)
{
	ToutI=ToutIC;
}
double TMc2::dame_ToutI(void)
{
	return (ToutI);
}
void TMc2::pon_ef_poli(double ef_poliC)
{
	ef_poli=ef_poliC;
}
double TMc2::dame_ef_poli(void)
{
	return (ef_poli);
}
void TMc2::pon_ef_isen(double ef_isenC)
{
	ef_isen=ef_isenC;
}
double TMc2::dame_ef_isen(void)
{
```

```
	return (ef_isen);
}
void TMc2::pon_Min(double MinC)
{
	Min=MinC;
}
double TMc2::dame_Min(void)
{
	return (Min);
}
void TMc2::pon_Din(double DinC)
{
	Din=DinC;
}
double TMc2::dame_Din(void)
{
	return (Din);
}
void TMc2::pon_Mout(double MoutC)
{
	Mout=MoutC;
}
double TMc2::dame_Mout(void)
{
	return (Mout);
}
void TMc2::pon_Dout(double DoutC)
{
	Dout=DoutC;
}
double TMc2::dame_Dout(void)
{
	return (Dout);
}
void TMc2::pon_min(double minC)
{
	min=minC;
}
double TMc2::dame_min(void)
{
	return (min);
}
void TMc2::pon_mout(double moutC)
{
	mout=moutC;
}
double TMc2::dame_mout(void)
{
	return (mout);
}
void TMc2::pon_Ain(double AinC)
{
	Ain=AinC;
}
double TMc2::dame_Ain(void)
{
	return (Ain);
}
void TMc2::pon_Aout(double AoutC)
{
	Aout=AoutC;
}
```

```
double TMc2::dame_Aout(void)
{
	return (Aout);
}

TMc2::TMc2(void)
{
	pon_nombre("Inicializado");
	pon_Pin(150000); //Pa
	pon_Tin(500); //k
	pon_Pout(600000); //Pa
	pon_Tout(600); //k
	pon_rc(3);
	pon_W(150000);  // J/kg
	pon_WI(150000);  // J/kg
	pon_ToutI(600); //k
	pon_ef_poli(0.85);
	pon_ef_isen(0.85);
	pon_Min(0.1);
	pon_Din(1);
	pon_Mout(0.5);
	pon_Dout(0.1);
	pon_min(100); //kg/s
	pon_mout(100); //kg/s
	pon_Ain(0.2); //m^2
	pon_Aout(0.1); //m^2
}
double TMc2::Fx1(double ToutIC, double PoutC, double TinC, double PinC, double
gamma,TMc2 &TMcompresor2)
{
	double fx1;

	fx1=((ToutIC/TinC) - (pow((PoutC/PinC),((gamma-1)/gamma))));

	TMcompresor2.pon_Pin(PinC);
	TMcompresor2.pon_Tin(TinC);
	TMcompresor2.pon_Pout(PoutC);
	TMcompresor2.pon_ToutI(ToutIC);
	return fx1;
}

double TMc2::Fx2(double PoutC, double PinC, double rcC,TMc2 &TMcompresor2)
{
	double fx2;

	fx2= (rcC-(PoutC/PinC));

	TMcompresor2.pon_rc(rcC);
	TMcompresor2.pon_Pin(PinC);
	TMcompresor2.pon_Pout(PoutC);
	return fx2;
}

double TMc2::Fx3(double ToutC, double TinC,double WC, double Cp,TMc2
&TMcompresor2)
{
	double fx3;

	fx3=(1 -((Cp*(ToutC - TinC))/WC));

	TMcompresor2.pon_W(WC);
	TMcompresor2.pon_Tin(TinC);
```

```
	TMcompresor2.pon_Tout(ToutC);
	return fx3;
}

double TMc2::Fx4(double ToutIC, double TinC,double WIC, double Cp,TMc2
&TMcompresor2)
{
	double fx4;

	fx4=(1 -((Cp*(ToutIC - TinC))/WIC));

	TMcompresor2.pon_WI(WIC);
	TMcompresor2.pon_Tin(TinC);
	TMcompresor2.pon_ToutI(ToutIC);
	return fx4;
}

double TMc2::Fx5(double WC, double ef_isenC,double WIC, TMc2 &TMcompresor2)
{
	double fx5;

	fx5=(1 -((ef_isenC*WC)/WIC));

	TMcompresor2.pon_W(WC);
	TMcompresor2.pon_WI(WIC);
	TMcompresor2.pon_ef_isen(ef_isenC);
	return fx5;
}

double TMc2::Fx6(double ToutC, double PoutC, double TinC, double PinC, double
ef_poliC, double gamma,TMc2 &TMcompresor2)
{
	double fx6;

	fx6=((ToutC/TinC) - (pow((PoutC/PinC),((gamma-1)/(ef_poliC*gamma)))));

	TMcompresor2.pon_Pin(PinC);
	TMcompresor2.pon_Tin(TinC);
	TMcompresor2.pon_Pout(PoutC);
	TMcompresor2.pon_Tout(ToutC);
	TMcompresor2.pon_ef_poli(ef_poliC);
	return fx6;
}

double TMc2::Fx7(double DinC, double MinC, double gamma, double R, TMc2
&TMcompresor2)
{
	double fx7, exp;
	exp= ((-1*(gamma+1))/(gamma-1));

	fx7=((DinC*DinC) - ((MinC*MinC)*(gamma/R)*(pow( (1+(((gamma-
1)/2)*(MinC*MinC))) , exp))));

	TMcompresor2.pon_Din(DinC);
	TMcompresor2.pon_Min(MinC);
	return fx7;
}

double TMc2::Fx8(double DoutC, double MoutC, double gamma, double R, TMc2
&TMcompresor2)
{
	double fx8, exp;
```

```
	exp= ((-1*(gamma+1))/(gamma-1));

	fx8=((DoutC*DoutC) - ((MoutC*MoutC)*(gamma/R)*(pow( (1+(((gamma-
1)/2)*(MoutC*MoutC))) , exp))));

	TMcompresor2.pon_Dout(DoutC);
	TMcompresor2.pon_Mout(MoutC);
	return fx8;
}

double TMc2::Fx9(double TinC, double PinC, double DinC, double minC, double
AinC,TMc2 &TMcompresor2)
{
	double fx9;

	fx9=((DinC*DinC)-((TinC*minC*minC)/(PinC*PinC*AinC*AinC)));

	TMcompresor2.pon_Tin(TinC);
	TMcompresor2.pon_Pin(PinC);
	TMcompresor2.pon_Din(DinC);
	TMcompresor2.pon_min(minC);
	TMcompresor2.pon_Ain(AinC);
	return fx9;
}
double TMc2::Fx10(double ToutC, double PoutC, double DoutC, double moutC, double
AoutC,TMc2 &TMcompresor2)
{
	double fx10;

	fx10=((DoutC*DoutC)-((ToutC*moutC*moutC)/(PoutC*PoutC*AoutC*AoutC)));

	TMcompresor2.pon_Tout(ToutC);
	TMcompresor2.pon_Pout(PoutC);
	TMcompresor2.pon_Dout(DoutC);
	TMcompresor2.pon_mout(moutC);
	TMcompresor2.pon_Aout(AoutC);
	return fx10;
}
double TMc2::Fx11(double minC, double moutC, TMc2 &TMcompresor2)
{
	double fx11;
	fx11=(minC-moutC);
	TMcompresor2.pon_min(minC);
	TMcompresor2.pon_mout(moutC);
	return fx11;
}

int TMc2::guardar_datos(int err, TMc2 &TMcompresor2)  //guardar datos
{
	FILE*out;
	if (err==1)
		out=fopen("datos.txt", "w");
	else
		out=fopen("datos.txt", "a");

	if(out==NULL)
	{
		return(-1);
	}
	else
	{
		fprintf(out,"%s\n",TMcompresor2.dame_nombre());
```

```
            fprintf(out,"Pin %lf\nTin %lf\nPout %lf\nTout %lf\nrc %lf\nW
%lf\n", TMcompresor2.dame_Pin(), TMcompresor2.dame_Tin(),
TMcompresor2.dame_Pout(), TMcompresor2.dame_Tout(),
TMcompresor2.dame_rc(),TMcompresor2.dame_W());
            fprintf(out,"Machin %lf\nMachout %lf\n", TMcompresor2.dame_Min(),
TMcompresor2.dame_Mout());
            fprintf(out,"Din %lf\nDout %lf\n", TMcompresor2.dame_Ain(),
TMcompresor2.dame_Aout());
            fprintf(out,"min %lf\nmout %lf\n", TMcompresor2.dame_min(),
TMcompresor2.dame_mout());
            fprintf(out,"Ain %lf\nAout %lf\n", TMcompresor2.dame_Ain(),
TMcompresor2.dame_Aout());
            fprintf(out,"Ef_poli %lf\nEf_isen %lf\n",
TMcompresor2.dame_ef_poli(), TMcompresor2.dame_ef_isen());
            fclose(out);
            return(0);
      }
}
```

Turbinas:

Modelización 0:

TMt0.h

```
#define MAX_ROW 20
#define MAX_COLUMN 20
#define MAXP 20
typedef char Tpalabra [MAXP];

class TMt0{
private:
      Tpalabra nombre_turbina;
      double Tout,Pout, Tin, Pin, rc, W, effc;
public:
      void pon_nombre (char nombre_turbina[20]);
      Tpalabra* dame_nombre (void);

      void pon_Tout (double ToutT);
      double dame_Tout (void);

      void pon_Pout (double PoutT);
      double dame_Pout (void);

      void pon_Tin (double TinT);
      double dame_Tin (void);

      void pon_Pin (double PinT);
      double dame_Pin (void);

      void pon_rc (double rcT);
      double dame_rc (void);

      void pon_W (double WT);
      double dame_W (void);

      TMt0(void);
      double Fx1(double ToutT, double PoutT, double TinT, double PinT, double
gamma,TMt0 &TMturbina0);
```

```
        double Fx2(double PoutT, double PinT, double rcT,TMt0 &TMturbina0);
        double Fx3(double ToutT, double TinT, double WT, double Cp,TMt0
&TMturbina0);
        int guardar_datos(int err, TMt0 &TMturrbina0);

};
```

TMt0.cpp

```
#include <stdio.h>
#include <string.h>
#include <math.h>
#include "TMt0.h"

void TMt0::pon_nombre(Tpalabra nombre2)
{
        strcpy(nombre_turbina,nombre2);
}
Tpalabra* TMt0:: dame_nombre(void)
{
        return (&nombre_turbina);
}
void TMt0::pon_Tout(double ToutT)
{
        Tout=ToutT;
}
double TMt0::dame_Tout(void)
{
        return (Tout);
}
void TMt0::pon_Pout(double PoutT)
{
        Pout=PoutT;
}
double TMt0::dame_Pout(void)
{
        return (Pout);
}
void TMt0::pon_Tin(double TinT)
{
        Tin=TinT;
}
double TMt0::dame_Tin(void)
{
        return (Tin);
}
void TMt0::pon_Pin(double PinT)
{
        Pin=PinT;
}
double TMt0::dame_Pin(void)
{
        return (Pin);
}
void TMt0::pon_rc(double rcT)
{
        rc=rcT;
}
double TMt0::dame_rc(void)
{
        return (rc);
```

```
}
void TMt0::pon_W(double WT)
{
	W=WT;
}
double TMt0::dame_W(void)
{
	return (W);
}
TMt0::TMt0(void)
{
	pon_nombre("Inicializado");
	pon_Pin(400000); //Pa
	pon_Tin(1200); //k
	pon_Pout(150000); //Pa
	pon_Tout(500); //k
	pon_rc(0.3);
	pon_W(150000); //J/Kg
}
double TMt0 ::Fx1(double ToutT, double PoutT, double TinT, double PinT, double
gamma,TMt0 &TMturbina0)
{
	double fx1;

	fx1=(pow (PinT,(1-gamma)) * pow (TinT,gamma))- (pow(PoutT,(1-gamma))* pow
(ToutT,gamma));

	TMturbina0.pon_Pin(PinT);
	TMturbina0.pon_Tin(TinT);
	TMturbina0.pon_Pout(PoutT);
	TMturbina0.pon_Tout(ToutT);
	return fx1;
}
double TMt0::Fx2(double PoutT, double PinT, double rcT,TMt0 &TMturbina0)
{
	double fx2;

	fx2= (PoutT-(rcT*PinT));

	TMturbina0.pon_rc(rcT);
	TMturbina0.pon_Pin(PinT);
	TMturbina0.pon_Pout(PoutT);
	return fx2;
}
double TMt0::Fx3(double ToutT, double TinT,double WT, double Cp,TMt0
&TMturbina0)
{
	double fx3;

	fx3=(WT -(Cp*(ToutT - TinT)));

	TMturbina0.pon_W(WT);
	TMturbina0.pon_Tout(ToutT);
	TMturbina0.pon_Tin(TinT);
	return fx3;
}
int TMt0::guardar_datos(int err, TMt0 &TMturbina0)  //guardar datos
{
	FILE*out;
	if (err==1)
		out=fopen("datos.txt", "w");
	else
```

```
		out=fopen("datos.txt", "a");

	if(out==NULL)
	{
		return(-1);
	}
	else
	{
		fprintf(out,"%s\n",TMturbina0.dame_nombre());
		fprintf(out,"%lf\n%lf\n%lf\n%lf\n%lf\n%lf\n",
TMturbina0.dame_Pin(), TMturbina0.dame_Tin(), TMturbina0.dame_Pout(),
TMturbina0.dame_Tout(), TMturbina0.dame_rc(),TMturbina0.dame_W());

		fclose(out);
		return(0);
	}
}
```

Modelización 1:

TMt1.h

```
#define MAX_ROW 20
#define MAX_COLUMN 20
#define MAXP 20
typedef char Tpalabra [MAXP];

class TMt1{
private:
	Tpalabra nombre_turbina;
	double Tout,Pout, Tin, Pin, rc, W, ToutI, WI, ef_poli, ef_isen;
public:

	void pon_nombre (char nombre_turbina[20]);
	Tpalabra* dame_nombre (void);

	void pon_Tout (double ToutT);
	double dame_Tout (void);

	void pon_Pout (double PoutT);
	double dame_Pout (void);

	void pon_Tin (double TinT);
	double dame_Tin (void);

	void pon_Pin (double PinT);
	double dame_Pin (void);

	void pon_rc (double rcT);
	double dame_rc (void);

	void pon_W (double WT);
	double dame_W (void);

	void pon_ToutI (double ToutIT);
	double dame_ToutI (void);

	void pon_WI (double WIT);
	double dame_WI (void);

	void pon_ef_isen (double ef_isenT);
```

```
        double dame_ef_isen (void);

        void pon_ef_poli (double ef_poliT);
        double dame_ef_poli (void);

        TMt1(void);
        double Fx1(double ToutIC, double PoutC, double TinC, double PinC, double
gamma,TMt1 &TMturbina1);
        double Fx2(double PoutC, double PinC, double rcC,TMt1 &TMturbina1);
        double Fx3(double ToutC, double TinC, double WC, double Cp,TMt1
&TMturbina1);
        double Fx4(double ToutIC, double TinC, double WIC, double Cp,TMt1
&TMturbina1);
        double Fx5(double WC, double ef_isenC, double WIC, TMt1 &TMturbina1);
        double Fx6(double ToutC, double PoutC, double TinC, double PinC, double
ef_poliC, double gamma,TMt1 &TMturbina1);
        int guardar_datos(int err, TMt1 &TMturbina1);
};
```

TMt1.cpp

```
#include <stdio.h>
#include <string.h>
#include <math.h>
#include "TMt1.h"


void TMt1::pon_nombre(Tpalabra nombre2)
{
        strcpy(nombre_turbina,nombre2);
}
Tpalabra* TMt1:: dame_nombre(void)
{
        return (&nombre_turbina);
}
void TMt1::pon_Tout(double ToutT)
{
        Tout=ToutT;
}
double TMt1::dame_Tout(void)
{
        return (Tout);
}
void TMt1::pon_Pout(double PoutT)
{
        Pout=PoutT;
}
double TMt1::dame_Pout(void)
{
        return (Pout);
}
void TMt1::pon_Tin(double TinT)
{
        Tin=TinT;
}
double TMt1::dame_Tin(void)
{
        return (Tin);
}
void TMt1::pon_Pin(double PinT)
```

```
{
	Pin=PinT;
}
double TMt1::dame_Pin(void)
{
	return (Pin);
}
void TMt1::pon_rc(double rcT)
{
	rc=rcT;
}
double TMt1::dame_rc(void)
{
	return (rc);
}
void TMt1::pon_W(double WT)
{
	W=WT;
}
double TMt1::dame_W(void)
{
	return (W);
}
void TMt1::pon_WI(double WIT)
{
	WI=WIT;
}
double TMt1::dame_WI(void)
{
	return (WI);
}
void TMt1::pon_ToutI(double ToutIT)
{
	ToutI=ToutIT;
}
double TMt1::dame_ToutI(void)
{
	return (ToutI);
}
void TMt1::pon_ef_poli(double ef_poliT)
{
	ef_poli=ef_poliT;
}
double TMt1::dame_ef_poli(void)
{
	return (ef_poli);
}
void TMt1::pon_ef_isen(double ef_isenT)
{
	ef_isen=ef_isenT;
}
double TMt1::dame_ef_isen(void)
{
	return (ef_isen);
}

TMt1::TMt1(void)
{
	pon_nombre("Inicializado");
	pon_Pin(400000); //Pa
	pon_Tin(600); //k
	pon_Pout(150000); //Pa
```

```
	pon_Tout(500); //k
	pon_rc(0.3);
	pon_W(150000);  // J/kg
	pon_WI(150000);
	pon_ToutI(500);
	pon_ef_poli(0.85);
	pon_ef_isen(0.85);
}
double TMt1::Fx1(double ToutIT, double PoutT, double TinT, double PinT, double
gamma,TMt1 &TMturbina1)
{
	double fx1;

	fx1=(pow (PinT,(1-gamma)) * pow (TinT,gamma))- (pow(PoutT,(1-gamma))* pow
(ToutIT,gamma));

	TMturbina1.pon_Pin(PinT);
	TMturbina1.pon_Tin(TinT);
	TMturbina1.pon_Pout(PoutT);
	TMturbina1.pon_ToutI(ToutIT);
	return fx1;
}

double TMt1::Fx2(double PoutT, double PinT, double rcT,TMt1 &TMturbina1)
{
	double fx2;

	fx2= (PoutT-(rcT*PinT));

	TMturbina1.pon_rc(rcT);
	TMturbina1.pon_Pin(PinT);
	TMturbina1.pon_Pout(PoutT);
	return fx2;
}

double TMt1::Fx3(double ToutT, double TinT,double WT, double Cp,TMt1
&TMturbina1)
{
	double fx3;

	fx3=(WT -(Cp*(ToutT - TinT)));

	TMturbina1.pon_W(WT);
	TMturbina1.pon_Tin(TinT);
	TMturbina1.pon_Tout(ToutT);
	return fx3;
}

double TMt1::Fx4(double ToutIT, double TinT,double WIT, double Cp,TMt1
&TMturbina1)
{
	double fx4;

	fx4=(WIT -(Cp*(ToutIT - TinT)));

	TMturbina1.pon_WI(WIT);
	TMturbina1.pon_Tin(TinT);
	TMturbina1.pon_ToutI(ToutIT);
	return fx4;
}

double TMt1::Fx5(double WT, double ef_isenT,double WIT, TMt1 &TMturbina1)
```

```
{
	double fx5;

	fx5=(WT -(ef_isenT*WIT));

	TMturbina1.pon_W(WT);
	TMturbina1.pon_WI(WIT);
	TMturbina1.pon_ef_isen(ef_isenT);
	return fx5;
}

double TMt1::Fx6(double ToutT, double PoutT, double TinT, double PinT, double
ef_poliT, double gamma,TMt1 &TMturbina1)
{
	double fx6;

	fx6=(pow (PinT,(1-gamma)) * pow (TinT,(gamma/ef_poliT)))- (pow(PoutT,(1-
gamma))* pow (ToutT,(gamma/ef_poliT)));

	TMturbina1.pon_Pin(PinT);
	TMturbina1.pon_Tin(TinT);
	TMturbina1.pon_Pout(PoutT);
	TMturbina1.pon_Tout(ToutT);
	TMturbina1.pon_ef_poli(ef_poliT);
	return fx6;
}

int TMt1::guardar_datos(int err, TMt1 &TMturbina1)  //guardar datos
{
	FILE*out;
	if (err==1)
		out=fopen("datos.txt", "w");
	else
		out=fopen("datos.txt", "a");

	if(out==NULL)
	{
		return(-1);
	}
	else
	{
		fprintf(out,"%s\n",TMturbina1.dame_nombre());
		fprintf(out,"%lf\n%lf\n%lf\n%lf\n%lf\n%lf\n",
TMturbina1.dame_Pin(), TMturbina1.dame_Tin(), TMturbina1.dame_Pout(),
TMturbina1.dame_Tout(), TMturbina1.dame_rc(),TMturbina1.dame_W());

		fclose(out);
		return(0);
	}
}
```

Modelización 2:

TMt2.h

```
#define MAX_ROW 20
#define MAX_COLUMN 20
#define MAXP 20
typedef char Tpalabra [MAXP];

class TMt2{
private:
	Tpalabra nombre_turbina;
	double Tout,Pout, Tin, Pin, rc, W, ToutI, WI, ef_poli, ef_isen, Min, Din,
Mout, Dout, min, mout, Ain, Aout ;
public:
	void pon_nombre (char nombre_turbina[20]);
	Tpalabra* dame_nombre (void);

	void pon_Tout (double ToutT);
	double dame_Tout (void);

	void pon_Pout (double PoutT);
	double dame_Pout (void);

	void pon_Tin (double TinT);
	double dame_Tin (void);

	void pon_Pin (double PinT);
	double dame_Pin (void);

	void pon_rc (double rcT);
	double dame_rc (void);

	void pon_W (double WT);
	double dame_W (void);

	void pon_ToutI (double ToutIT);
	double dame_ToutI (void);

	void pon_WI (double WIT);
	double dame_WI (void);

	void pon_ef_isen (double ef_isenT);
	double dame_ef_isen (void);

	void pon_ef_poli (double ef_poliT);
	double dame_ef_poli (void);

	void pon_Min (double MinT);
	double dame_Min (void);

	void pon_Din (double DinT);
	double dame_Din (void);

	void pon_Mout (double MoutT);
	double dame_Mout (void);

	void pon_Dout (double DoutT);
	double dame_Dout (void);

	void pon_min (double minT);
	double dame_min (void);

	void pon_mout (double moutT);
	double dame_mout (void);
```

```
        void pon_Ain (double AinT);
        double dame_Ain (void);

        void pon_Aout (double AoutT);
        double dame_Aout (void);

        TMt2(void);
        double Fx1(double ToutIT, double PoutT, double TinT, double PinT, double
gamma,TMt2 &TMturbina2);
        double Fx2(double PoutT, double PinT, double rcT,TMt2 &TMturbina2);
        double Fx3(double ToutT, double TinT, double WT, double Cp,TMt2
&TMturbina2);
        double Fx4(double ToutIT, double TinT, double WIT, double Cp,TMt2
&TMturbina2);
        double Fx5(double WT, double ef_isenT, double WIT, TMt2 &TMturbina2);
        double Fx6(double ToutT, double PoutT, double TinT, double PinT, double
ef_poliT, double gamma,TMt2 &TMturbina2);

        double Fx7(double DinT, double MinT, double gamma, double R,TMt2
&TMturbina2);
        double Fx8(double DoutT, double MoutT, double gamma, double R,TMt2
&TMturbina2);
        double Fx9(double TinT, double PinT, double DinT, double minT, double
AinT,TMt2 &TMturbina2);
        double Fx10(double ToutT, double PoutT, double DoutT, double moutT,
double AoutT,TMt2 &TMturbina2);
        double Fx11(double minT, double moutT, TMt2 &TMturbina2);
        int guardar_datos(int err, TMt2 &TMturbina2);
};
```

TMt2.cpp

```
#include <stdio.h>
#include <string.h>
#include <math.h>
#include "TMt2.h"

void TMt2::pon_nombre(Tpalabra nombre2)
{
        strcpy(nombre_turbina,nombre2);
}
Tpalabra* TMt2:: dame_nombre(void)
{
        return (&nombre_turbina);
}
void TMt2::pon_Tout(double ToutT)
{
        Tout=ToutT;
}
double TMt2::dame_Tout(void)
{
        return (Tout);
}
void TMt2::pon_Pout(double PoutT)
{
        Pout=PoutT;
}
double TMt2::dame_Pout(void)
{
        return (Pout);
```

```
}
void TMt2::pon_Tin(double TinT)
{
	Tin=TinT;
}
double TMt2::dame_Tin(void)
{
	return (Tin);
}
void TMt2::pon_Pin(double PinT)
{
	Pin=PinT;
}
double TMt2::dame_Pin(void)
{
	return (Pin);
}
void TMt2::pon_rc(double rcT)
{
	rc=rcT;
}
double TMt2::dame_rc(void)
{
	return (rc);
}
void TMt2::pon_W(double WT)
{
	W=WT;
}
double TMt2::dame_W(void)
{
	return (W);
}
void TMt2::pon_WI(double WIT)
{
	WI=WIT;
}
double TMt2::dame_WI(void)
{
	return (WI);
}
void TMt2::pon_ToutI(double ToutIT)
{
	ToutI=ToutIT;
}
double TMt2::dame_ToutI(void)
{
	return (ToutI);
}
void TMt2::pon_ef_poli(double ef_poliT)
{
	ef_poli=ef_poliT;
}
double TMt2::dame_ef_poli(void)
{
	return (ef_poli);
}
void TMt2::pon_ef_isen(double ef_isenT)
{
	ef_isen=ef_isenT;
}
double TMt2::dame_ef_isen(void)
```

```
{
	return (ef_isen);
}
void TMt2::pon_Min(double MinT)
{
	Min=MinT;
}
double TMt2::dame_Min(void)
{
	return (Min);
}
void TMt2::pon_Din(double DinT)
{
	Din=DinT;
}
double TMt2::dame_Din(void)
{
	return (Din);
}
void TMt2::pon_Mout(double MoutT)
{
	Mout=MoutT;
}
double TMt2::dame_Mout(void)
{
	return (Mout);
}
void TMt2::pon_Dout(double DoutT)
{
	Dout=DoutT;
}
double TMt2::dame_Dout(void)
{
	return (Dout);
}
void TMt2::pon_min(double minT)
{
	min=minT;
}
double TMt2::dame_min(void)
{
	return (min);
}
void TMt2::pon_mout(double moutT)
{
	mout=moutT;
}
double TMt2::dame_mout(void)
{
	return (mout);
}

void TMt2::pon_Ain(double AinT)
{
	Ain=AinT;
}
double TMt2::dame_Ain(void)
{
	return (Ain);
}
void TMt2::pon_Aout(double AoutT)
{
```

```
	Aout=AoutT;
}
double TMt2::dame_Aout(void)
{
	return (Aout);
}
TMt2::TMt2(void)
{
	pon_nombre("Inicializado");
	pon_Pin(400000); //Pa
	pon_Tin(600); //k
	pon_Pout(150000); //Pa
	pon_Tout(500); //k
	pon_rc(0.3);
	pon_W(150000);  // J/kg
	pon_WI(9000);
	pon_ToutI(500);
	pon_ef_poli(0.85);
	pon_ef_isen(0.85);
	pon_Min(0.5);
	pon_Din(0.1);
	pon_Mout(0.5);
	pon_Dout(1);
	pon_min(100); //kg/s
	pon_mout(100); //kg/s
	pon_Ain(0.1); //m^2
	pon_Aout(0.2); //m^2
}
double TMt2::Fx1(double ToutIT, double PoutT, double TinT, double PinT, double
gamma,TMt2 &TMturbina2)
{
	double fx1;

	fx1=(pow (PinT,(1-gamma)) * pow (TinT,gamma))- (pow(PoutT,(1-gamma))* pow
(ToutIT,gamma));

	TMturbina2.pon_Pin(PinT);
	TMturbina2.pon_Tin(TinT);
	TMturbina2.pon_Pout(PoutT);
	TMturbina2.pon_ToutI(ToutIT);
	return fx1;
}

double TMt2::Fx2(double PoutT, double PinT, double rcT,TMt2 &TMturbina2)
{
	double fx2;

	fx2= (PoutT-(rcT*PinT));

	TMturbina2.pon_rc(rcT);
	TMturbina2.pon_Pin(PinT);
	TMturbina2.pon_Pout(PoutT);
	return fx2;
}

double TMt2::Fx3(double ToutT, double TinT,double WT, double Cp,TMt2
&TMturbina2)
{
	double fx3;

	fx3=(WT -(Cp*(ToutT - TinT)));
```

```
	TMturbina2.pon_W(WT);
	TMturbina2.pon_Tin(TinT);
	TMturbina2.pon_Tout(ToutT);
	return fx3;
}

double TMt2::Fx4(double ToutIT, double TinT,double WIT, double Cp,TMt2
&TMturbina2)
{
	double fx4;

	fx4=(WIT -(Cp*(ToutIT - TinT)));

	TMturbina2.pon_WI(WIT);
	TMturbina2.pon_Tin(TinT);
	TMturbina2.pon_ToutI(ToutIT);
	return fx4;
}

double TMt2::Fx5(double WT, double ef_isenT,double WIT, TMt2 &TMturbina2)
{
	double fx5;

	fx5=(WT -(ef_isenT*WIT));

	TMturbina2.pon_W(WT);
	TMturbina2.pon_WI(WIT);
	TMturbina2.pon_ef_isen(ef_isenT);
	return fx5;
}

double TMt2::Fx6(double ToutT, double PoutT, double TinT, double PinT, double
ef_poliT, double gamma,TMt2 &TMturbina2)
{
	double fx6;

	fx6=(pow (PinT,(1-gamma)) * pow (TinT,(gamma/ef_poliT)))- (pow(PoutT,(1-
gamma))* pow (ToutT,(gamma/ef_poliT)));

	TMturbina2.pon_Pin(PinT);
	TMturbina2.pon_Tin(TinT);
	TMturbina2.pon_Pout(PoutT);
	TMturbina2.pon_Tout(ToutT);
	TMturbina2.pon_ef_poli(ef_poliT);
	return fx6;
}

double TMt2::Fx7(double DinT, double MinT, double gamma, double R, TMt2
&TMturbina2)
{
	double fx7, exp;
	exp= ((-1*(gamma+1))/(gamma-1));

	fx7=((DinT*DinT) - ((MinT*MinT)*(gamma/R)*(pow( (1+(((gamma-
1)/2)*(MinT*MinT))) , exp))));

	TMturbina2.pon_Din(DinT);
	TMturbina2.pon_Min(MinT);
	return fx7;
}

double TMt2::Fx8(double DoutT, double MoutT, double gamma, double R, TMt2
```

```
&TMturbina2)
{
	double fx8, exp;
	exp= ((-1*(gamma+1))/(gamma-1));

	fx8=((DoutT*DoutT) - ((MoutT*MoutT)*(gamma/R)*(pow( (1+(((gamma-
1)/2)*(MoutT*MoutT))) , exp))));

	TMturbina2.pon_Dout(DoutT);
	TMturbina2.pon_Mout(MoutT);
	return fx8;
}

double TMt2::Fx9(double TinT, double PinT, double DinT, double minT, double
AinT,TMt2 &TMturbina2)
{
	double fx9;

	fx9=((DinT*DinT)-((TinT*minT*minT)/(PinT*PinT*AinT*AinT)));

	TMturbina2.pon_Tin(TinT);
	TMturbina2.pon_Pin(PinT);
	TMturbina2.pon_Din(DinT);
	TMturbina2.pon_min(minT);
	TMturbina2.pon_Ain(AinT);
	return fx9;
}
double TMt2::Fx10(double ToutT, double PoutT, double DoutT, double moutT, double
AoutT,TMt2 &TMturbina2)
{
	double fx10;

	fx10=((DoutT*DoutT)-((ToutT*moutT*moutT)/(PoutT*PoutT*AoutT*AoutT)));

	TMturbina2.pon_Tout(ToutT);
	TMturbina2.pon_Pout(PoutT);
	TMturbina2.pon_Dout(DoutT);
	TMturbina2.pon_mout(moutT);
	TMturbina2.pon_Aout(AoutT);
	return fx10;
}
double TMt2::Fx11(double minT, double moutT, TMt2 &TMturbina2)
{
	double fx11;
	fx11=(minT-moutT);
	TMturbina2.pon_min(minT);
	TMturbina2.pon_mout(moutT);
	return fx11;
}
int TMt2::guardar_datos(int err, TMt2 &TMturbina2)  //guardar datos
{
	FILE*out;
	if (err==1)
		out=fopen("datos.txt", "w");
	else
		out=fopen("datos.txt", "a");

	if(out==NULL)
	{
		return(-1);
	}
	else
```

```
	{
		fprintf(out,"%s\n",TMturbina2.dame_nombre());
		fprintf(out,"Pin %lf\nTin %lf\nPout %lf\nTout %lf\nrc %lf\nW
%lf\n", TMturbina2.dame_Pin(), TMturbina2.dame_Tin(), TMturbina2.dame_Pout(),
TMturbina2.dame_Tout(), TMturbina2.dame_rc(),TMturbina2.dame_W());
		fprintf(out,"Machin %lf\nMachout %lf\n", TMturbina2.dame_Min(),
TMturbina2.dame_Mout());
		fprintf(out,"Din %lf\nDout %lf\n", TMturbina2.dame_Ain(),
TMturbina2.dame_Aout());
		fprintf(out,"min %lf\nmout %lf\n", TMturbina2.dame_min(),
TMturbina2.dame_mout());
		fprintf(out,"Ain %lf\nAout %lf\n", TMturbina2.dame_Ain(),
TMturbina2.dame_Aout());
		fprintf(out,"Ef_poli %lf\nEf_isen %lf\n",
TMturbina2.dame_ef_poli(), TMturbina2.dame_ef_isen());
		fclose(out);
		return(0);
	}
}
```

Fan:

Modelización 2:

TMf2.h

```
#include "TMc2.h"
#define MAX_ROW 35
#define MAX_COLUMN 35
#define MAXP 20
typedef char Tpalabra [MAXP];

class TMf2{
private:
	Tpalabra nombre_fan;
	double min, Ain, B, W;
public:
	void pon_nombre (char nombre_fan[20]);
	Tpalabra* dame_nombre (void);

	void pon_min (double minF);
	double dame_min (void);

	void pon_Ain (double AinF);
	double dame_Ain (void);

	void pon_B (double BF);
	double dame_B (void);

	void pon_W (double WF);
	double dame_W (void);

	TMc2 compresor_interno;
	TMc2 compresor_externo;
	TMf2(void);
	double Fx1(double minF, double min_intF, double min_extF, TMf2 &TMfan2);
	double Fx2(double AinF, double Ain_intF, double Ain_extF,TMf2 &TMfan2);
	double Fx3(double BF, double min_intF, double min_extF,TMf2 &TMfan2);
	double Fx4(double WF, double W_intF, double W_extF,TMf2 &TMfan2);
```

```
};
```

TMf2.cpp

```
#include <stdio.h>
#include <string.h>
#include <math.h>
#include "TMf2.h"

void TMf2::pon_nombre(Tpalabra nombre2)
{
	strcpy(nombre_fan,nombre2);
}
Tpalabra* TMf2:: dame_nombre(void)
{
	return (&nombre_fan);
}
void TMf2::pon_min(double minF)
{
	min=minF;
}
double TMf2::dame_min(void)
{
	return (min);
}
void TMf2::pon_Ain(double AinF)
{
	Ain=AinF;
}
double TMf2::dame_Ain(void)
{
	return (Ain);
}
void TMf2::pon_B(double BF)
{
	B=BF;
}
double TMf2::dame_B(void)
{
	return (B);
}
void TMf2::pon_W(double WF)
{
	W=WF;
}
double TMf2::dame_W(void)
{
	return (W);
}

TMf2::TMf2(void)
{
	pon_nombre("Inicializado");
	pon_min(20);  //kg/s
	pon_Ain(0.2); //m^2
	pon_B(3);
```

```
	pon_W(350000);
}

double TMf2::Fx1(double minF, double min_intF, double min_extF, TMf2 &TMfan2)
{
	double fx1;

	fx1= (minF-(min_intF+min_extF));

	TMfan2.pon_min(minF);
	TMfan2.compresor_interno.pon_min(min_intF);
	TMfan2.compresor_externo.pon_min(min_extF);
	return fx1;
}

double TMf2::Fx2(double AinF, double Ain_intF, double Ain_extF,TMf2 &TMfan2)
{
	double fx2;

	fx2= (AinF-(Ain_intF+Ain_extF));

	TMfan2.pon_Ain(AinF);
	TMfan2.compresor_interno.pon_Ain(Ain_intF);
	TMfan2.compresor_externo.pon_Ain(Ain_extF);
	return fx2;
}

double TMf2::Fx3(double BF, double min_intF, double min_extF,TMf2 &TMfan2)
{
	double fx3;

	fx3= ((BF*min_intF)-min_extF);

	TMfan2.pon_B(BF);
	TMfan2.compresor_interno.pon_min(min_intF);
	TMfan2.compresor_externo.pon_min(min_extF);
	return fx3;
}

double TMf2::Fx4(double WF, double W_intF, double W_extF,TMf2 &TMfan2)
{
	double fx4;

	fx4= (WF-(W_intF+W_extF));

	TMfan2.pon_W(WF);
	TMfan2.compresor_interno.pon_W(W_intF);
	TMfan2.compresor_externo.pon_W(W_extF);
	return fx4;
}
```

Ejemplo de banco de pruebas: (Banco de Pruevas TMc0.cpp)

```
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include <math.h>
#include "f2c.h"
#include "clapack.h"
#include "Newton.h"
#include "TMc0.h"
#define MAX_ROW 20
#define MAX_COLUMN 20
#define MAX_PAR 20

void main()
{
	TMc0 compresor;
	Newton MNewton;
	Tpalabra nombre;
	int op=-1, i=-1, p=0, nFx=-1, longitud, j, s, err=1;
	int C[MAX_ROW][MAX_COLUMN];
	double h, rho, gamma =1.4, Cp=1005;
	double TinC, ToutC, PinC, PoutC, rcC, WC;
	integer n=nFx+1, incx=1, incy=1;
	doublereal da;
	double delta_x=0.00001;
	double Xaux[MAX_ROW+MAX_PAR], *X=&Xaux[MAX_PAR], Fx [MAX_ROW], Jaux
[MAX_ROW], Jx [MAX_ROW][MAX_COLUMN];

	while (op != 0)
	{

		printf ("\n		Seleccione una opcion:\n");
		printf ("		0.Salir del programa\n");
		printf ("		1.Introducir datos al compresor\n");
		printf ("		2.Solucionar ensamblaje\n");
		printf ("		3.Mostrar componentes\n");
		scanf ("%d", &op);
		switch (op)
		{
		case 0:
			printf ("\nGracias por usar el programa\n\n");
			system ("pause");
			break;

		case 1:
			compresor;
			i=-1;
			p=0;
			printf("\n	ID. COMPRESOR: ");
			flushall();		//para evitar errores
			scanf("%s",nombre);
			longitud=strlen(nombre);
			if(longitud>MAXP)
			{
				printf("El nombre no puede tener mas de %d
caracteres\n",MAXP);
				break;
			}
			compresor.pon_nombre(nombre);
			printf ("\nDetermine el valor de las variables conocidas y
asigne un -1 a las desconocidas:\n");

			printf ("Temperatura a la salida(k):");
```

```
scanf ("%lf", &ToutC);

if (ToutC==-1)
{
	i++;
	C[0][0]=i;
	C[2][0]=i;
	X[i]=compresor.dame_Tout();
}
else
{
	p--;
	C[0][0]=p;
	C[2][0]=p;
	X[p]=ToutC;
	compresor.pon_Tout(ToutC);
}

printf ("Presion a la salida(Pa):");
scanf ("%lf", &PoutC);

if (PoutC==-1)
{
	i++;
	C[0][1]=i;
	C[1][0]=i;
	X[i]=compresor.dame_Pout();
}
else
{
	p--;
	C[0][1]=p;
	C[1][0]=p;
	X[p]=PoutC;
	compresor.pon_Pout(PoutC);
}

printf ("Temperatura a la entrada(k):");
scanf ("%lf", &TinC);

if (TinC==-1)
{
	i++;
	C[0][2]=i;
	C[2][1]=i;
	X[i]=compresor.dame_Tin();
}
else
{
	p--;
	C[0][2]=p;
	C[2][1]=p;
	X[p]=TinC;
	compresor.pon_Tin(TinC);
}

printf ("Presion a la entrada(Pa):");
scanf ("%lf", &PinC);

if (PinC==-1)
{
	i++;
```

```
                        C[0][3]=i;
                        C[1][1]=i;
                        X[i]=compresor.dame_Pin();
                }
                else
                {
                        p--;
                        C[0][3]=p;
                        C[1][1]=p;
                        X[p]=PinC;
                        compresor.pon_Pin(PinC);
                }

                printf ("Relacion de compresion:");
                scanf ("%lf", &rcC);

                if (rcC==-1)
                {
                        i++;
                        C[1][2]=i;
                        X[i]=compresor.dame_rc();
                }
                else
                {
                        p--;
                        C[1][2]=p;
                        X[p]=rcC;
                        compresor.pon_rc(rcC);
                }

                printf ("Trabajo del compresor(J/kg):");
                scanf ("%lf", &WC);

                if (WC==-1)
                {
                        i++;
                        C[2][2]=i;
                        X[i]=compresor.dame_W();
                }
                else
                {
                        p--;
                        C[2][2]=p;
                        X[p]=WC;
                        compresor.pon_W(WC);
                }

                err=compresor.guardar_datos(err, compresor);

                break;

        case 2:
                nFx=0;
                Fx[nFx] = compresor.Fx1(X[C[nFx][0]], X[C[nFx][1]],
X[C[nFx][2]], X[C[nFx][3]], gamma,compresor);
                nFx++;
                Fx[nFx] = compresor.Fx2(X[C[nFx][0]], X[C[nFx][1]],
X[C[nFx][2]],compresor);
                nFx++;
                Fx[nFx] = compresor.Fx3(X[C[nFx][0]], X[C[nFx][1]],
X[C[nFx][2]],Cp,compresor);
```

```
                    // Es crea la matriz de derivadas parciales
                    j=0;
                    while (j <= i)
                    {
                           X[j] = X[j] + delta_x;

                           nFx=0;
                           Jaux[nFx] = compresor.Fx1(X[C[nFx][0]], X[C[nFx][1]],
X[C[nFx][2]], X[C[nFx][3]], gamma,compresor);
                           nFx++;
                           Jaux[nFx] = compresor.Fx2(X[C[nFx][0]], X[C[nFx][1]],
X[C[nFx][2]],compresor);
                           nFx++;
                           Jaux[nFx] = compresor.Fx3(X[C[nFx][0]], X[C[nFx][1]],
X[C[nFx][2]],Cp,compresor);

                           X[j] = X[j]- delta_x; //se restablece el valor

                           n=nFx+1;
                           da=-1;
                           daxpy_(&n, &da, Fx, &incx, Jaux, &incy);
      //Resta de vectores

                           da=(1/delta_x);
                           dscal_(&n, &da, Jaux, &incx);
      //division por un enter

                           s=0;
                           while (s <= nFx)
                           {
                           Jx[s][j]=Jaux[s]; // Se cololcan las derivadas
parciales en el lugar correspondiente del jacobiano.
                           s++;
                           }

                           j++;
                    }
                    MNewton.Desarrollo(X,Fx,Jx, s, nFx);
                    double nrm2;
                    n=nFx+1;
                    nrm2 = dnrm2_(&n, Fx, &incx);   // norma 2
                    while ((nrm2<=0.00001)==false)
                    {
                           nFx=0;
                           Fx[nFx] = compresor.Fx1(X[C[nFx][0]], X[C[nFx][1]],
X[C[nFx][2]], X[C[nFx][3]], gamma,compresor);
                           nFx++;
                           Fx[nFx] = compresor.Fx2(X[C[nFx][0]], X[C[nFx][1]],
X[C[nFx][2]],compresor);
                           nFx++;
                           Fx[nFx] = compresor.Fx3(X[C[nFx][0]], X[C[nFx][1]],
X[C[nFx][2]],Cp,compresor);

                           j=0;
                           while (j <= i)
                           {
                                  X[j] = X[j] + delta_x;

                                  nFx=0;
                                  Jaux[nFx] = compresor.Fx1(X[C[nFx][0]],
X[C[nFx][1]], X[C[nFx][2]], X[C[nFx][3]], gamma,compresor);
                                  nFx++;
```

```
                                    Jaux[nFx] = compresor.Fx2(X[C[nFx][0]],
X[C[nFx][1]], X[C[nFx][2]],compresor);
                                    nFx++;
                                    Jaux[nFx] = compresor.Fx3(X[C[nFx][0]],
X[C[nFx][1]], X[C[nFx][2]],Cp,compresor);

                                    X[j] = X[j]- delta_x; //es restableix el valor

                                    n=nFx+1;
                                    da=-1;
                                    daxpy_(&n, &da, Fx, &incx, Jaux, &incy);
             //Resta de vectors
                                    da=(1/delta_x);
                                    dscal_(&n, &da, Jaux, &incx);
      //divisio per un enter

                                    s=0;
                                    while (s <= nFx)
                                    {
                                    Jx[s][j]=Jaux[s]; // Se cololcan las derivadas
parciales en el lugar correspondiente del jacobiano.
                                    s++;
                                    }
                                    j++;
                             }

                             MNewton.Desarrollo(X,Fx,Jx, s, nFx);
                             err=compresor.guardar_datos(err, compresor);

                             n=nFx+1;
                             nrm2 = dnrm2_(&n, Fx, &incx);
                      }

                      break;

              case 3:
                      printf("\n    COMPRESOR : %s\n ", compresor.dame_nombre());
                      printf(" Temperatura a la entrada:       %lf    k\n ",
compresor.dame_Tin());
                      printf(" Presion a la entrada:           %lf    Pa\n ",
compresor.dame_Pin());
                      printf(" Temperatura a la salida: %lf    k\n ",
compresor.dame_Tout());
                      printf(" Presion a la salida:            %lf    Pa\n ",
compresor.dame_Pout());
                      printf(" Relacion de compresion:  %lf\n ",
compresor.dame_rc());
                      printf(" Trabajo del compresor :  %lf    J/kg\n ",
compresor.dame_W());
                      break;
              }
      }
}
```